13. SVM

Tuesday, January 12, 2021 5:03 PM

SVM לא עובד עם יותר מ2 קטגוריות

SVM נבנה עבור קלסיפיקציה בינארית ולא ניתן להרחבה בקלות ליותר מ2 קטגוריות

1. One VS One
   - בנה k(k-1)/2 קלסיפיירים : אחד לכל זוג קטגוריות
   - בהינתן דוגמא x הפעל את כל הקלסיפייר
   - פלט החיזוי: הקטגוריה בעלת רוב החיזויים
2. One VS. Rest
   - בנה k קלסיפיירים שמבדילים אם קלט x שייך לקטגוריה i או לכל יתר הקטגוריות
   - פלט החיזוי: הקלסיפייר שנותן את הביטחון המקסימלי לקטגוריה שלו: $h_i(x)$ המקסימלי

SVM

מאוד מסובך בפנים

נשתמש בו בקופסא שחורה לרוב

משתמש ב3 Kernels עקריים
Linear, Polynomial, Gaussian

מתי נרצה להשתמש

כאשר רוצים לבדוק מהר גבולות החלטה לא ליניאריים

כשאין לנו הרבה data
ז.א חוששים מ -High Variance
מפחדים מ Over fitting

כשהבעיה ממד גבוהה
A lot of parameters
High Dimension

אבל: קימות בעיות ביצועים כאשר D גדול מאוד
מטריצת ערכי K (הקרנלים) היא ריבועית במספר הדוגמאות m
אם כי, ישנן ואריאציות עבור קרנלים מסויימים שהמטריצה K לא תהיה ענקית

1. Support Vector Classifier (Soft Max Margin)
   - Better generalization ability & Controlling bias-variance tradeoff
   - Can be extended for regression
   - Accessible and easy to use
2. The Kernel Trick
   - Map data points to higher dimensional space in order to make them linearly separable.
   - Since only dot product is used, no need to represent the mapping explicitly.
   - Kernel Trick can be adapted to many parametric learning algorithms
3. Not a miraculous algorithm:
   - Not very efficient to train when D is large
   - Not very efficient to use at inference when number of support vectors is large
   - Practically, when dimensionality is high, there are many support vectors
   - Does NOT extend easily to multi-class
   - Similar to other well established algorithms:
     - Logistic regression (possible with kernels)
     - Weighted-KNN)

Hinge Loss with L2 Regularization

בעזרת לגרנגיאן ניתן להפוך את האילוצים לפונקצית loss ולהראות כי בעית האופטימיזציה הקוודרטית שקולה למיזעור הפונקציה:

$$minimize_w \left\{ \sum_{i=1}^{m} \max[0,1 - y_i h(x_i)] + \gamma \sum_{j=1}^{n} {w_j}^2 \right\}$$

Hinge loss - פונקצית loss שממזערת חריגות מ M=1.
- כש yh(x) גדולה מ1 - ה loss הוא 0.
- אם x בתוך ה margin, הloss בין 0-1.
- אם x בצד השני של גבול ההחלטה, loss>1

Ridge Regularization:
Penalty γ controls bias-variance
in the same way as C

תחת האילוצים הרגילים:

$for\ every\ \ (x_i, yi) \in D$
$y_i\,(wx_i) \geq\ (\mathbf{1} - \boldsymbol{\varepsilon}_\mathbf{i})$
$\varepsilon_i\ \geq 0,$
$\sum_{i=1}^{m} \varepsilon_i < C$

מסקנה: SVM שקול לביצוע GD על hinge עם רגורליזציה L2.
(בהנחה ש GD אכן ימצא מינימום גלובאלי)

90

SVM מתנהג בדומה לכמה אלגוריתמי למידה אחרים:
1. דוגמה לרגרסיה לוגיסטית עם גרולריזצית ridge עם אותו הkernel
2. דומה לKNN כשמשתמשים בRadial Kernel (Gaussian)

סביר שיהיה במבחן

SVM לינארי הוא כמעט זהה לרגרסיה לוגיסטית עם רגולריזציה

אם מחליפים את ה loss <-
במקום CrossEntropy נשתמש ב Hinge אז אפשר לעשות GD על פונקציית ה loss שכוללת את הRidge + Hinge loss ולקבל תוצאות זהות לSVM

SVM זהה ל Gradient Descent אם נשנה את הloss ל Hinge ונשתמש ב Ridge

Support Vector Machine (SVM)

אלגוריתם פארמטרי שימושי ונגיש מאוד: הומצא ע"י תיאורטיקנים בשנות ה 90,

נראה היה מסתורי, שונה ואפקטיבי מכל אלגוריתם אחר עד לאותה תקופה:

- גרם לזניחה כמעט טוטאלית של רשתות נוירונים בשנות ה 90.
- לאט לאט התברר שהאלגוריתם אקוויולנטי לטכניקות ידועות ב ML:

Support Vector Machine is a binary classifier

נבדיל במינוחים

- Maximum Margin Classifier

  קלסיפייר בינארי לינארי על בסיס מיקסום המרווח (margin) בין הקטגוריות - דורש כי הנתונים יהיו ניתנים להפרדה לינארית (בדומה לפרספטרון)

- Support Vector Classifier

  הרחבה לקלסיפייר בינארי לינארי שאינו דורש הפרדה לינארית מוחלטת, ומאפשר גם outliers.
  פארמטר Capacity מאפשר שליטה על Bias-variance tradeof

- SVM:

  הרחבה לקלסיפייר בינארי לא לינארי ע"י טריק הקרנל

הHinge הוא דואג שהMargin יהיה 1 קבוע, אז אם המרחק של נקודה מסוימת גדול מהmargin (1) אז אין שגיאה (נבחר 0 ב MAX)

מה באמת (t) חיצוי (y)

$$\max[0,1 - y_i h(x_i)]$$

כתוב יפה בכחול בתמונה (משאיר כי אולי אתקן בהמשך)
אם חזינו שלילי, אבל באמת הוא חיובי, אז $y_i h(x_i)$ שלילי כנ"ל הפוך.
במקרה הזה ה max יבחר $1 - (-num) > \mathbf{1}$

אם נקודות **מחוץ ל margin בצד הנכון**: הmax יבחר בין 0 לבין הוודאות של הנקודה (הביטוי $y_i h(x_i)$ יהיה גדול מ1 אז יבחר 0)

אם נקודה היא **בתוך הmargin ובצד הנכון** (הביטוי $y_i h(x_i)$ יהיה בין 0 ל1 אז יבחר)

אם זה בתוך הmargin אבל בצד הלא נכון (של גבול ההחלטה) השגיאה היא גדולה מ1

סה"כ נמדוד את השגיאות ונמזער אותן בgradient descent

Why [-1 , +1] ? Wil help us with the math later..

0:07:00

Maximal Margin Classifier:
היפר-מישור מפריד c

בהינתן קבוצת אימון D של דוגמאות מסווגות בינארית {+1,-1} ממימד n, הניתנת להפרדה לינארית,
רוצים למצוא היפר-מישור ממימד n-1 המפריד בין הדוגמאות

אנחנו מניחים שD ניתן להפרדה לינארית

במימד n ההיפר-מישור המפריד הוא אילוץ מהצורה :

$$h(x)=w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = 0$$

למשל: במימד 2, למישור המפריד צורה של קו לינארי $w_0 + w_1x_1 + w_2x_2 = 0$

כל נקודה שמקיימת את האילוץ, היא נקודה על המישור המפריד

נקודות מעל למישור יסווגו כ 1 : $w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n > 0$

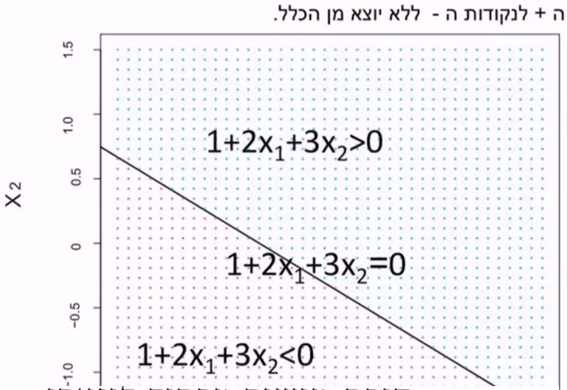
נקודות מתחת למישור יסווגו כ -1 : $w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n < 0$

הסיווג של X הוא לכן : $\mathbf{sign}(w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n)$

נשים לב שפה אין הסתברות(בניגוד ל__________), יש רק חיזוי (לפי ה sign של ההיפותזה (+ או -))

Hyper-plan 1+2x1+3x2=0

D במימד n ניתן להפרדה לינארית אם ניתן להעביר היפר מישור במימד n-1 המפריד בין נקודות ה + לנקודות ה - ללא יוצא מן הכלל.



דוגמה בשני ממדים

היפר-מישור מפריד לינארי

בהינתן קבוצת אימון D של דוגמאות $x_i = (x_{i1}, x_{i2}, \ldots x_{in})$
מסווגות בינארית $y_i \in\{+1,-1\}$
הניתנת להפרדה לינארית,

למישור המפריד יש את התכונה:

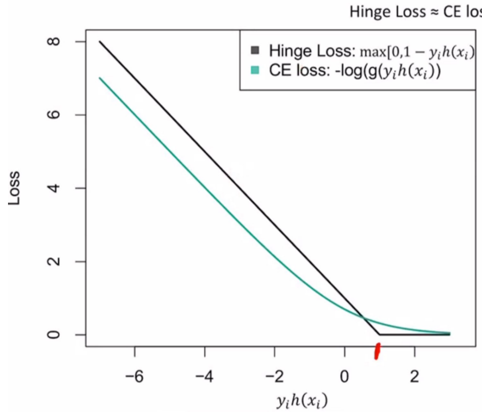
$$y_i\, h(x_i) = y_i (w_0 + w_1x_{i1} + w_2x_{i2} + \ldots + w_nx_{in}) > 0$$

המרחק של נקודה x מהמישור נותן אינדיקציה לרמת הביטחון בסיווג:

$$\frac{y_i\,(w_0 + w_1x_{i1} + w_2x_{i2} + \ldots + w_nx_{in})}{\sqrt{\sum_{j=1}^{n} {w_j}^2}}$$

קשר הדוק בין SVM לרגרסיה לוגיסטית

Hinge Loss ≈ CE loss



0:17:00

היפר-מישור מפריד לינארי

בהינתן קבוצת אימון D של דוגמאות $x_i = (x_{i1}, x_{i2}, \dots x_{in})$

מסווגות בינארית $y_i \in \{+1,-1\}$

הניתנת להפרדה לינארית,

למישור המפריד יש את התכונה:

$$y_i\, h(x_i) = y_i(w_0 + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_n x_{in}) > 0$$

אם נכפיל את החיזוי בתוצאה של ההיפותזה נקבל תמיד משהו שגדול מ-0

כי אם זה גדול מ0 אז סווגנו את זה ב+

אם זה קטן מ0 סיווגנו ב-

ההפלה תצא חיובית בכל מקרה

המרחק של נקודה x מהמישור נותן אינדיקציה לרמת הביטחון בסיווג:

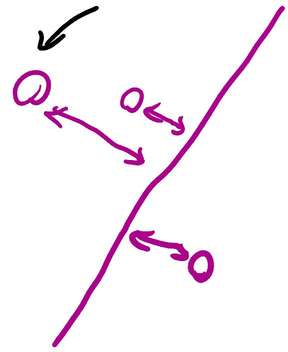
$$\frac{y_i\,(w_0 + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_n x_{in})}{\sqrt{\sum_{j=1}^{n} w_j^{\,2}}}$$

מרחק מנקודה למישור

הכפלה שנועדה כדי לקבל מספר חיובי

חלוקה בנורמה

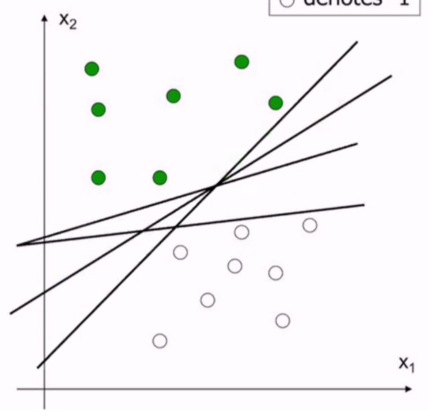
יותר בטוח



ישנם אין סוף מישורים המקיימים את האילוצים (לכל נקודה). באיזה נבחר?

ברגרסיה לוגיסטית בחרנו מישור מפריד שממזער את ה CE, עבור SVM נפעיל לוגיקה אחרת

היפר-מישור מפריד לינארי



- Which one is the best?

למקסם את הmargin <-

למקסם את המרחק לנקודות הכי קרובות

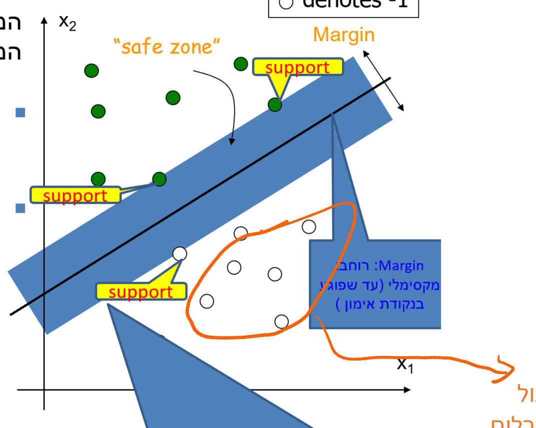
למה?

זה מוריד Variance

Large Margin Linear Classifier

המישור המפריד בעל ה Margin המקסימלי, הוא הטוב ביותר!

- ה Margin מוגדר כרוחב שבו ניתן להרחיב את ההיפר-מישור המפריד לפני שנתקל בנקודת אימון
- מדוע המישור המפריד עם ה margin המקסימלי הוא הטוב ביותר?
- יכולת הכללה טובה יותר עבור נקודות טסט המתקרבות למישור המפריד
- אינו רגיש למיקום רוב הנקודות שמחוץ ל margin ולכן פחות Overfitting



- כשממקסימים את ה margin, רק היפר-מישור אחד יכול להיות עם margin מקסימלי

איך בוחרים את הקו?

נמקסם את רוחב המלבן

למקסם את הmargin <-

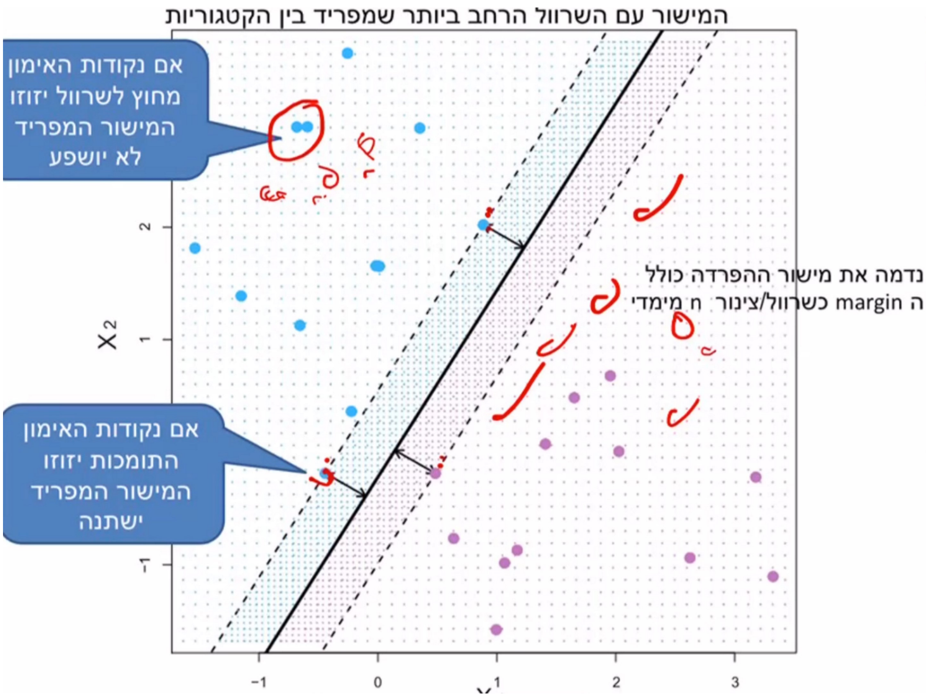
למקסם את המרחק לנקודות הכי קרובות

למה?

זה מוריד Variance

הנקודות הרחוקות מהקו בכלל לא משפיעות על גבול ההחלטה, הן יכולות להשנות ולהתווסף וזה לא יזיז כלום

איך בוחרים את השרוול של הSVM



בניית Max Margin Classifier:
בעיית אופטימיזציה קוודרטית

המטרה:
למצוא את המשקולות w
שימקסמו את M

- $Maximize \quad M$
  $M(w_0, w_1, w_2, , , , , w_n), w_i \text{ are the params that effect } M$

האילוץ,
המרחק של כל הנקודות בD ממרכז הצינור
יהיה **גדול מ M**

- subject to $y_i\,(w_0 + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_n x_{in}) \geq M$
  $\text{for every} \quad (x_i, y_i) \in D$

- Subject to $\sum_{j=1}^{n} {w_j}^2 = 1$

ישנם הרבה מישור מפרידים אקוולנטים
(כל הכפלת משוואת המישור בקבוע).

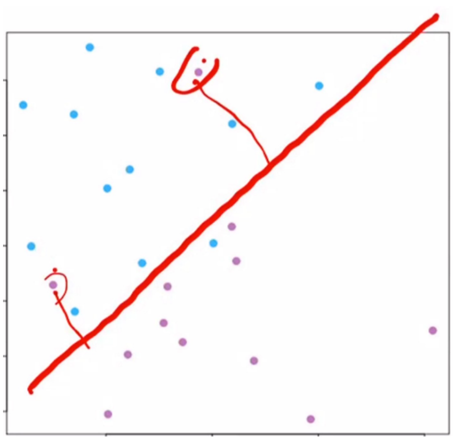
- בוחר מבין כל המישורים כזה עם נורמל ליחידה,
- נותן משמעות ל $w_0 + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_n x_{in}$: המרחק של הנקודה מהמישור
- הבעיה מנוסחת כך שהפיתרון הוא גלובאלי ויחיד

לדוגמה:
2x+5
הוא אותו מישור כמו
4x+10
ויש אינסוף מישורים כאלה
(כל ההכפלות)

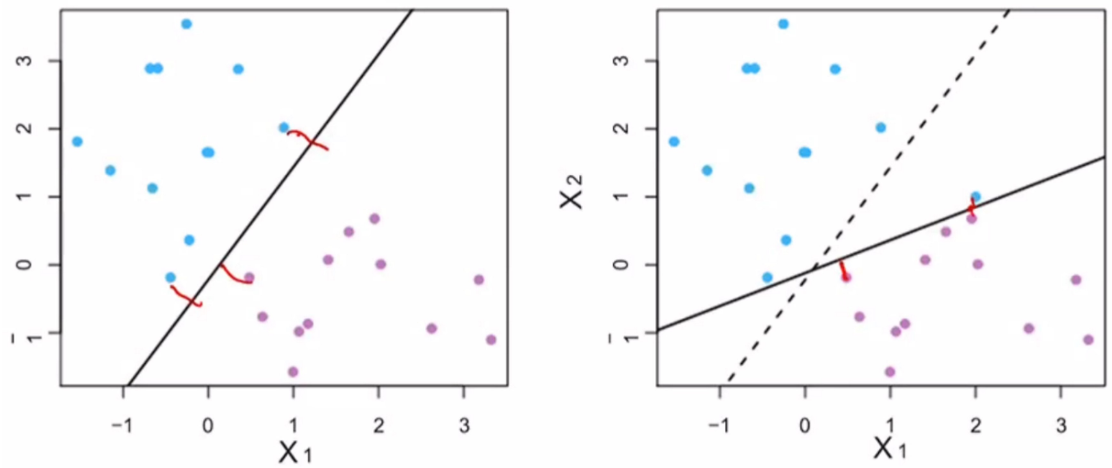
33:30
בחלק מהספרות מתארים את הבעיה באופן הבא:
אנחנו מחפשים Margin יחיד, ואנחנו מחפשים את המשקולות המינימאליות
[שומרים על margin אחד וממזערים את סכום המשקולות]

0:34:00

כשנקודות אינן ניתנות להפרדה לינארית, אין פיתרון
לבעיית האופטימיזציה.
Max Margin Classifier פשוט לא קיים



תוספת של נקודה אחת עלולה לגרום לתזוזה דרסטית של המישור המפריד
ול margin צר מאוד- אינדיקציה ל Over-fitting



Max Margin Classifier - Problems

- אם הנתונים לא ניתנים להפרדה לינארית,
 אין פתרון לבעיית האופטימיזציה.
 הגדרת הבעיה אינה מאפשרת outliers בכלל!
- גם אם ניתן להפריד, **ישנה רגישות גבוה לנקודות שבחזית** (בסמוך למישור המפריד). שינוי במיקום וקטורי התמיכה, כמו תוספת נקודות בגבולות ה margin יכול לגרום לשינוי דרסטי במישור המפריד ול Margin צר מאוד

0:44:00

Support Vector (Soft) Classifier - לא דורש הפרדה לינארית מלאה

נחפש **Soft** Margin Classifier שאינו מכריח כל דוגמא להיות מעבר ל margin

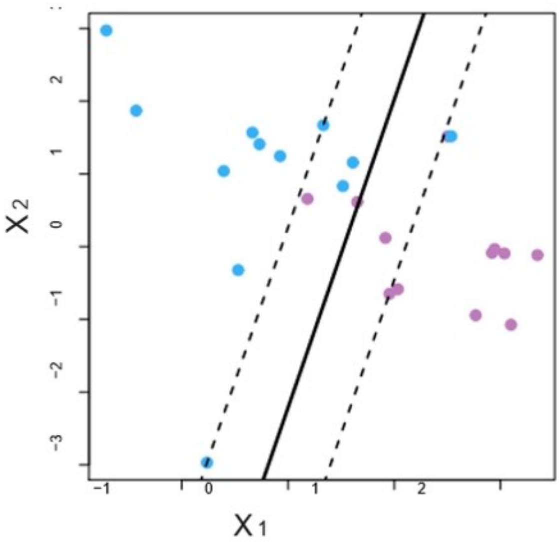
כיצד?

- *נאפשר: מיעוט דוגמאות תוכלנה להופיע בצד הלא נכון של ה margin מבלי להצר אותו*
- *נאפשר: מיעוט דוגמאות תוכלנה אפילו להופיע בצד הלא נכון של מישור ההפרדה מבלי שישפיעו על מישור ההפרדה ואפילו על ה margin (ז.א. לא נדרוש הפרדה לינארית מושלמת)*

על מנת לבצע קלסיפיקציה טובה בנוכחות outliers ורעש, רצוי לאפשר פרדיקציה שגויה לפעמים

נרוויח:

- *margin רחב יחד עם קלסיפיקציה טובה של רוב דוגמאות האימון*
- *סבילות Robustness לOutLiers בודדים*
- *נוכל לאפשר מספר רב יותר של נקודות support –*
התוצאה: פחות רגישות לתזוזות ונקודות חדשות



אינטואיציה:
כיצד מגדירים זאת כבעיית אופטימיזציה?:
רוצים למקסם את ה margine אבל עבור margin רחב מוכנים מוכנים "לשלם" שמיעוט נקודות יכנסו לתוך הצינור ואפילו יסווגו כשגויים

Support Vector Classifier = Soft Margin Classifier
נוסיף משתנים לבעיית האופטימיזציה

נוסיף Slack variable ε_i לכל דוגמה, אשר מאפשר לה להופיע בצד הלא נכון: ε_i הוא התשלום שמשלמים עבור חריגה מה Margin
(נרצה למזער את סכום התשלומים)

- כאשר $\varepsilon_i = 0$ הדוגמא , תהיה מחוץ לmargin (תסווג נכון) — ולא יתווסף דבר לסכום המוגבל ב C
- כאשר $0 < \varepsilon_i \leq 1$ הדוגמא תהיה בצד הלא נכון של ה margin אבל בצד הנכון של המישור (תסווג נכון)
- כאשר $\varepsilon_i > 1$ הדוגמא תהיה בצד הלא נכון של המישור המפריד (תסווג שגוי)

במקרים אלה יתווסף לתחום המוגבל בC

יהיה לנו תקציב C שאנחנו מוכנים לשלם עבור הנקודות החורגות הללו:
ככל שהחריגה גדולה יותר, התשלום εi גדול יותר.

נרצה להרחיב את הmargin כמה שיותר, אך ובשום פנים לא לחרוג מהתקציב

ככל שנגדיל את C הVariance יותר קטן
(כמו לעשות רגולריזציה)

כי אנחנו לוקחים Margin יותר גדול
ומספר הנקודות שתומכות יותר קטן,
ככה שגם אם אחת מהן תזוז מעט,
תיהיה השפעה גדולה

0:50:00

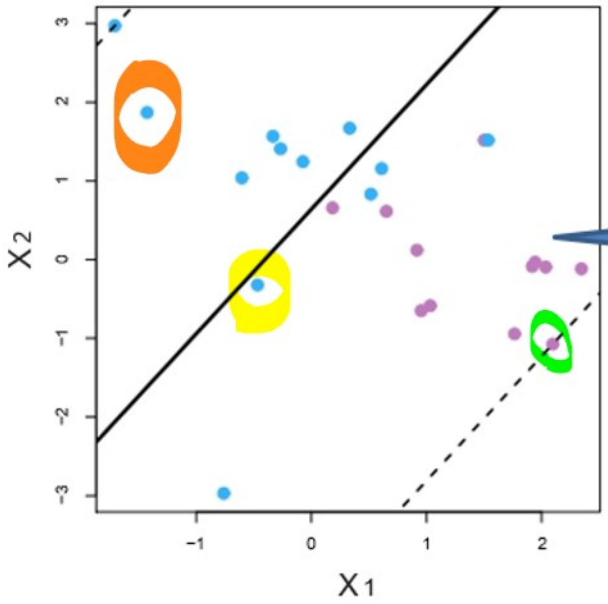
ניסוח בעיית האופטימיזציה

$$Maximize \quad M$$

$$M, w_0, w_1, w_2, \ldots, w_n, \boldsymbol{\varepsilon_i}, \ldots \boldsymbol{\varepsilon_m}$$

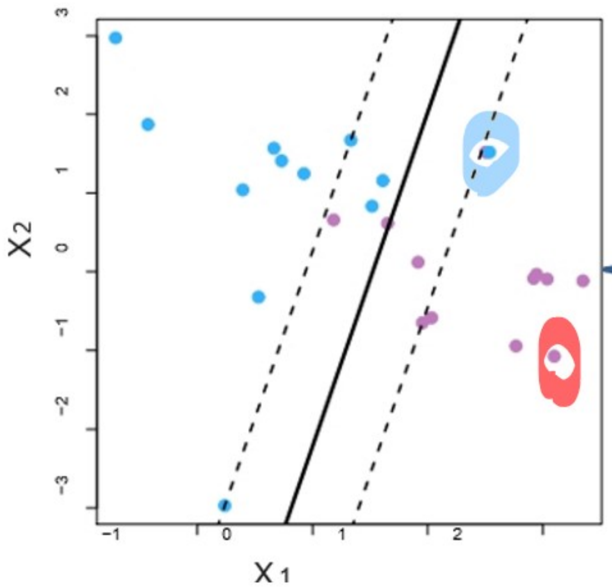
Subject to: $for\ every \quad (x_i, y_i) \in D$ המרחק של כל נקודה ב Training Set גדול מ

- $y_i\,(w_0 + w_1x_{i1} + w_2x_{i2} + \cdots + w_nx_{in}) \geq M(\mathbf{1} - \boldsymbol{\varepsilon_i})$
- $\boldsymbol{\varepsilon_i} \geq \mathbf{0},$
- $\sum_{i=1}^{m} \boldsymbol{\varepsilon_i} < \mathbf{C}$ סכום כל הSlackVars יהיה קטן מהתקציב C
- $\sum_{j=1}^{n} w_j^{\,2} = 1$ האילוץ על הנורמה שראינו קודם (שתהיה בגודל 1)



C גדול מאפשר marginרחב:
מספר גדול של נקודות תמיכה,
ווריאנס קטן

1. $\epsilon_{yellow} = 1^+$ בצד הלא נכון, לא רחוק מגבול ההחלטה
2. $\epsilon_{green} = 0^+$ בצד הנכון, טיפה בתוך הMargin
3. $\epsilon_{orange} = 0.4 \pm$ בצד הנכון, בערך באמצע



C קטן יאפשר רק margin צר,
מספר קטן של נקודות תמך,
ווריאנס גדול

1. $\epsilon_{\text{blue}} = 2^+$ בצד הלא נכון, יצא מהmargin אבל ממש קרוב אליו
2. $\epsilon_{\text{red}} = 0$ בצד הנכון, מחוץ לmargin

- C גבוה: מאפשר margin רחב הבנוי על הרבה וקטורי תמך
  - למרות שיהיו הרבה דוגמאות שיפרו את ה margin.
  - מספר וקטורי התמך גדול ולפיכך פחות רגישות לשינויים בוקטורי התמך (פחות שגיאת ווריאנס)
- C קטן, יצור margin צר, הבנוי על מעט וקטורי תמך.

1:03:00

נקודות התמך

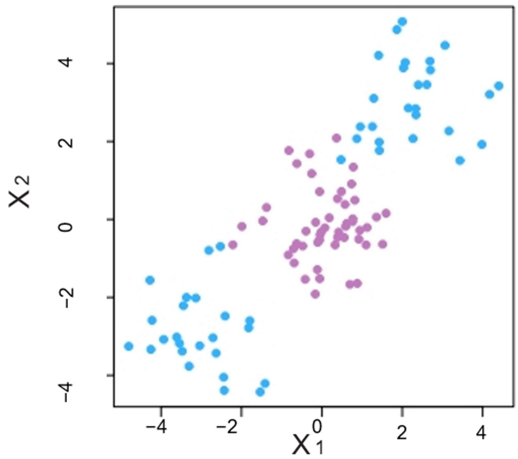
- נקודות מחוץ ל margin אינן משפיעות כלל על המישור המפריד. אפשר להוציאם, או להזיזם כל עוד לא הופכות לנקודות תמך מבלי לשנות את המישור ואת ה margin
- נקודות בתוך גבולות ה margin הם נקודות תמך:
  - שינוי בהם עלול לגרום לשינוי במישור המפריד וגם ב margin.
    (כולל נקודות שחצו את גבול ה margin לצד הלא נכון.)
  - אם מספר נקודות התמך קטן, שינוי קל במיקומם עלול להשפיע דרסטית, מה שמעיד על Over-fitting

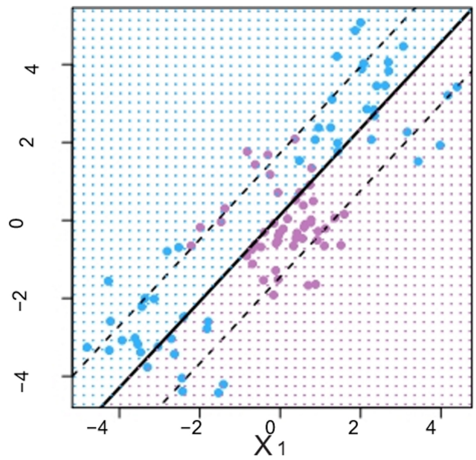
לפעמים נקראות גם: הנקודות שבחזית, או הנקודות "שמחזיקות" את היפר-צינור ה Margin

Kernels Linear, Polynomial , Gaussian

ביצד נתמודד עם גבולות לא ליניאריים?

SVC ימצא פתרון גרוע



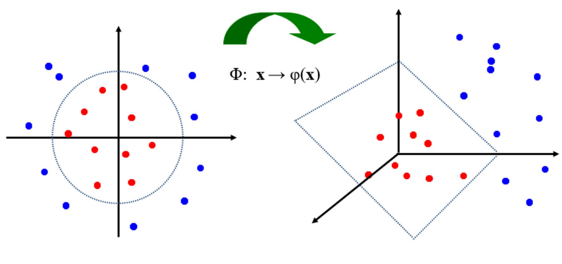


כזכור: הסבת מרחב ה feature (למשל: לקואורדינטות פולריות, או למרחב פולינומיאלי)
יאפשר גבולות החלטה לא ליניארים (על המרחב המקורי)

לSupport Vectors מצאו שיטה טובה (שמסתבר שעובדת טוב גם לאלגוריתמים אחרים) - KERNELS
הKernels עושים את אותו הדבר, רק בצורה יותר יעילה חישובית (מאשר להסב \ להעלות מימדים \ גאוסייניים)

- אפשר שנוסיף polynomial features:
  quadratic/cubic/k-order…
  ונבצע את האופטימיזציה במרחב ממימד גדול יותר
- אפשר להשתמש במאפינים לאו דווקא פולינומיאלים

הרעיון הכללי: ניתן למפות את מרחב הקלטים למרחב ממימד גבוה יותר שם הנתונים כן ניתנים להפרדה ליניארית



מספר האפשרויות עצום ואם לא ניזהר יהיו בעיות חישוב!.

SVM משתמש ב"טריק ה Kernel" שמאפשר הרחבת המרחב בו משתמש ה SV Classifier באופן שהוא "בהרבה מיקרים" יעיל יותר מבחינה חישובית
וללא המרה בפועל של הווקטורים

דוגמאות ל kernels פופולרים

1:12:00

קימים המוני קרנלים. כמעט לכל מטרה,
למשל קרנלים להשוואת דימיון בין גרפים....
אבל ה-3 הללו הם הכי פופולריים

Examples of Kernels

Linear: $K(x, x') = x \cdot x'$

Polynomial: $K(x, x') = (x \cdot x')^d$

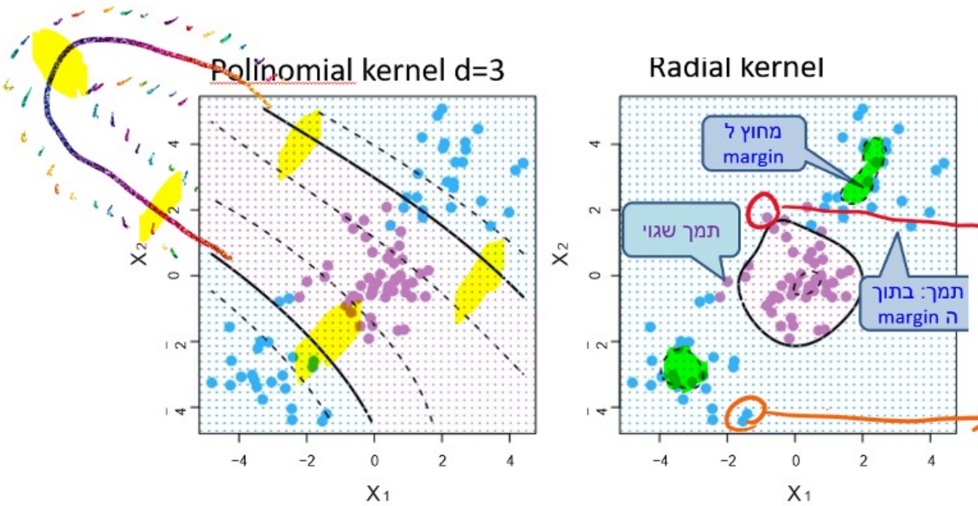
Gaussian: $K(x, x') = \exp(-\frac{1}{2}\|x - x'\|/\sigma)$

מדוע SVM עם linear kernel יהיה טוב יותר
מפרספטרון או מ naïve Bayes?
הרבה בעיות למידה היום – במימדים גבוהים.
למשל סיווג טקסטים.
כשהמימד גבוה, אפילו naive Bayes מתחיל
לעשות Overfitting (למשל כאשר מילה
אקראית הינה קורלטיבית עם סיווג מסוים)
SVM do not overfit easily ומתאים במיוחד
למימדים גדולים

Polinomial Kernel גם בצורתו הכי פשוטה
d=2 נותן הרבה כוח.

קרנל גאוסיאני: דומה ל RBF נותן דימיון בעל צורת פעמון.
מעלים בחזקה את הנורמה (מרחק אאוקלידי בריבוע) ומחלקים ברוחב
ה kernel.
ככל שהסיגמה קטנה, הגאוסיאן צר יותר (low-bias, High-variance)
ככל שהסיגמה גדולה,
פונקצית הדימיון הופכת להיות מרחק אאוקלידי
(High-bias, low variance)
סיגמה : היפר פארמטר

האם נראה שגאוסיאן כזה הוא מכפלה
סקלרית של 2 טרנספורמציות?
זיכרו שמספר הfeatures יכול לגדול
בטרנספורמציה :
אבל לא עושים בפועל את הטרנספורמציה-
מחשבים רק את K שמשמעותו היא כמו
מכפלה סקלרית של 2 טרנספורמציות



$\epsilon_{point} = 0$ בתוך השטח הירוק ה

הנקודה ה עם slack של גדול מ1
בערך כי היא בסיווג לא נכון

הנקודות האלה בתוך ה Margin ולכן
יש להן Slack שבטווח [0-1] כי הן
בצד הנכון של גבול ההחלטה

יתרונות חישוביים בKernel

- אין צורך לעבור למרחב הגדול. מספיק לחשב את הדימיון בין כל זוגות הנקודות (במקרה הגרוע $O(m^2)$)
  המעבר למרחב הגדול נעשה באופן לא מפורש implicit.
  יתרון גדול כי בהרבה אפליקציית המרחב הגדול הוא עצום
- בזמן אימון צריך לחשב את הדימיון של כל הזוגות אך בזמן חיזוי לנקודה x מספיק לחשב את הדימיון שלה רק לוקטורי התמך
- ה Kernel מאפשר לfeatures לעבור טרנספורמציה, אבל נשארת הפשטות והקלות של העבודה הלינארית אחרי הטרנספורמציה

(לפעמים המרחב המוגדל הוא **ממםד** אין-סופי ואז אין בכלל אפשרות להעביר את הבעיה למרחב הגדול בצורה מפורשת - כמו במקרה של radial kernels)

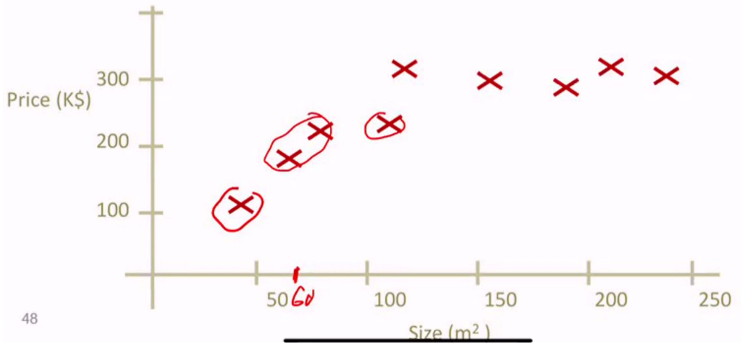
שיטה להוסיף דרגות גמישות למודל (מלחמה בUnder-Fitting)
במקום שנחזה קו לינארי לפי כל המידע שלנו (בזמן האימון), ונחזה ערך חדש לפיו.
נשתמש בשיטה הזו: Locally Weighted Linear Regression
נעביר כל פעם קו בהתאם לקרבה לנקודה החדשה.

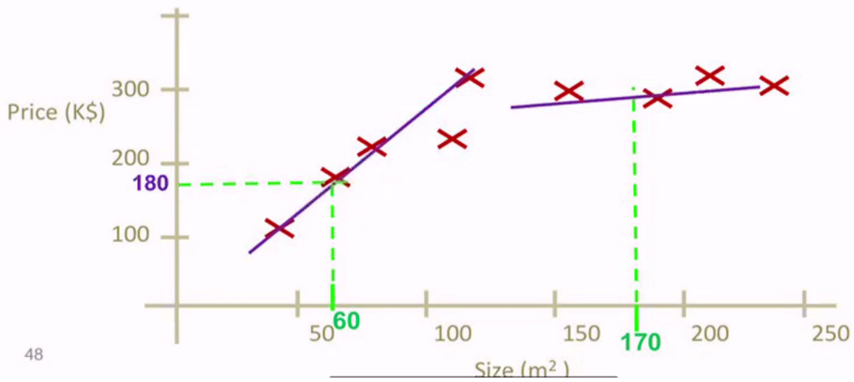
שליטה על הגמישות ברגרסיה לינארית:

- Locally Weighted Linear Regression
- לא נבנה מודל בזמן האימון, אלא נחכה ל query:

בהנתן Query (ווקטור מאפיינים X)

נבנה מודל רגרסיה כך שהנקודות ב D הקרובות יותר ל X ישפיעו על הרגרסיה יותר מהנקודות הרחוקות ממנו





קל באימונים - קשה בקרב

בזמן האימון רק שומרים בזכרון את הדוגמאות

בזמן החיזוי יוצרים קו רגרסיה לינארי מה הנקודות הקרובות ביותר לקלט

החסרונות הן
1. לוקח הרבה זמן להוציא פרדיקציה
2. לא מתאים לbig Data

הרחבה של LR: Locally weighted linear regression (LWR)

היבריד של שיטה לא פארמטרית ביחד עם רגרסיה לינארית לוקאלית

אימון: שומרים בזיכרון דוגמאות ומשתמשים בהם בכל פעם שזקוקים לחיזוי

חיזוי:

- בהנתן x מחשבים משקולות [0-1] לדוגמאות האימון על פי מרחקיהם מ x.
  דוגמה רחוקה: משקלה יתקרב ל 0
  דוגמה קרובה, משקלה יתקרב ל 1.
- מבצעים רגרסיה עם פונקציית MSE ממושקלת:

$$WMSE_w = \frac{1}{2m}\sum_i \beta_i (w x_i - t_i)^2$$

ה β היא משקל שאנחנו נותנים לכל בTraining set

בטא תלויה במרחק (המשקולל) בין הנקודה החדשה, לכל נקודה ב training set שלנו D

שגיאות בנקודות קרובות הן חשובות

שגיאות בנקודות רחוקות הן פחות חשובות

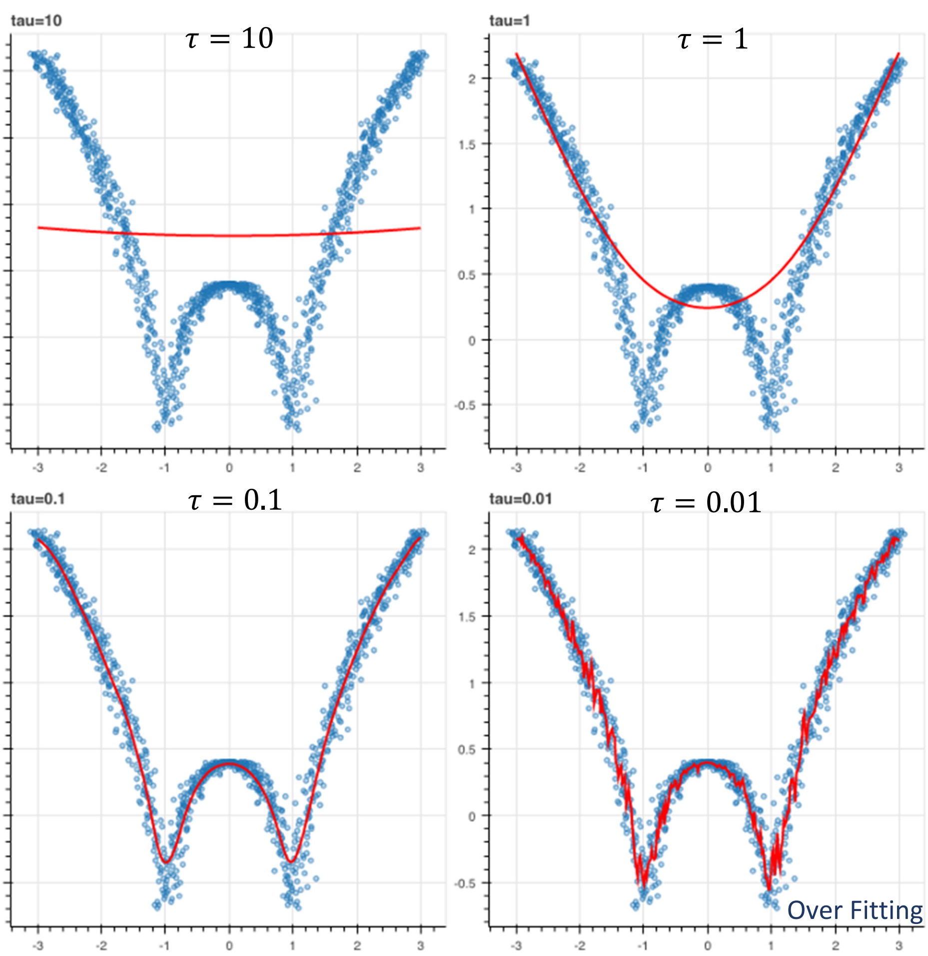
דוגמה לפונקצית דימיון:

מרחק מחושב ע"פ הפונקציה הגאוסיאנית מסביב לנקודה:

$$\beta_i = e^{\frac{-||x_i - x||^2}{2\tau^2}}$$

- ככל ש τ קטן: מרחקים גדולים מקבלים משקל אפסי (ורק נקודות קרובות מאוד משפיעות)
- ככל ש τ גדול יותר כך נלקחות בחשבון גם נקודות רחוקות

τ - שולטת על הגמישות



ניתן לשלוט על הגמישות
בעזרת פונקצית המשקל:

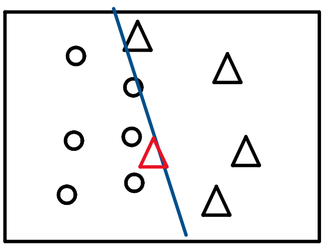
$$\beta i = e^{\frac{-||xi-x||^2}{2\tau^2}}$$

ככל שטאו קטן יותר
המשקל של נקודות
מרוחקות שואף ל 0.
והנקודה הכי קרובה
משפיעה הרבה.
כשטאו גדול מאוד כל
הנקודות משפיעות באופן
דומה ומקבלים רגרסיה
לינארית רגילה

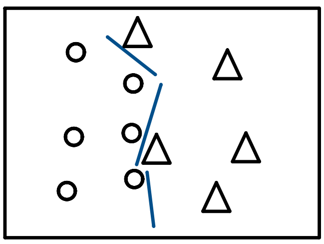
גרסה קיימת גם ל logistic reg LW
דיי דומה ל KNN

הערה:
את אותה טכניקה שמוסיפה גמישות אפשר לעשות לרגרסיה לוגיסטית (קלסיפיקציה).

אם יש לנו למשל את המצב הזה:



קו רגיל לא יצליח



אבל נעשה לפי נקודות קרובות אז נוכל להפריד

(לא כ"כ מקובל כי רגרסיה לוגיסטית צריכה GD
וGD יותר כבדה מהNormal Equasion, אז NE במימדים קטנים הוא מאוד מהיר, בעוד ש GD צריך לעשות הרבה איטרציות.
אז בגלל השיטה העצלנית של "קל באימונים, קשה בקרב", אנחנו נצטרך משהו מאוד מהיר בזמן query.