Machine Learning

- Supervised: **Labeled** examples

קיים מורה שנותן תווית רצויה לכל קלט (קלסיפיקציה או רגרסיה)
לומדים תבניות המקשרות בין המאפיינים לתוויות.

- Unsupervised: **Unlabeled** examples

לא קיים מורה : לומדים תבניות המתקיימות בין המאפיינים.

- Reinforcement: Unlabeled examples with a positive/negative teaching signal

קיים מורה שרק נותן חיווי : הצליח לא הצליח. לומדים לבחור פעולה שתמקסם רווח (תגמול) עתידי (לא בקורס).

- Semi supervised: most data are unlabeled.

Supervised:

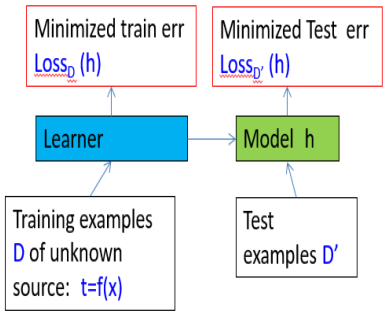
Step 1- Training: Building a prediction model out of labeled data.

Step 2 – Validating: Evaluate the learned model on Test data ("Home test").

Step 3 – Final Test: If the result is not enough then it is possible to return to previous steps.

Step 4 – Production

משימת החיזוי איננה קלה : נתונים דוגמאות מהעבר, אך רוצים לבצע חיזוי עבור דוגמאות חדשות שעדיין לא נראו. יכולת ההכללה, נמדדת בשלב הtest על דוגמאות מבחן שלא ניצפו בזמן האימון.



הבעיה :
נתונה קבוצת אימון $D = \{< \underline{X}_i, \underline{T}_{train} >\}$

וקבוצת מבחן $V = \{< \underline{X}_i, \underline{T}_{test} >\}$
רוצים ללמוד את הפונקציה h (היפותזה אופטימלית המייצגת מודל) :
$t=f(x)\approx h(x)$

כך ששגיאת החיזוי $loss_V(h)$ על דוגמאות המבחן תהיה מזערית.

$\underline{X}_i$ => vector of samples of feature i. (X is a vector of $\underline{X}_i$s)

$t_i = f(\underline{X}_i) + \varepsilon \approx h(\underline{X}_i) = y_i$ => label (real result) i, fits to $\underline{X}_i$ samples.
($\underline{T}$ is a vector of t_is). ε is a noise.

$h_w(x)=w_0+w_1x_1+\ldots w_nx_n$.

- השגיאה ε היא אינהרנטית (משתנה מיקרי) ומקורה ב : טעויות מדידה, קלט שאיננו מספיק עבור חישוב הפלט (ישנם אולי משתנים חשובים וחבויים שאינם ב D), אקראיות והרעש במערכות פיסיקליות.

- בד"כ בפרקטיקה : מניחים שהתוחלת של השגיאה ε מתאפסת. מניחים גם ש ε אינה תלויה סטטיסטית בקלט (לא חייב להתקיים במציאות).

- שיטות פרמטריות : מניחים ש $h_w(x)$ היא פונקציה פארמטרית מסוימת- **(המודל)** : מתאימים את המודל לנתונים ע"י פרוצדורת אימון שמוצא את הפרמטרים שימזערו את השגיאה.

- שיטות לא פרמטריות: לא מניחות f ספציפי אלא משתמשות במדגם כדי לבנות את הפונקציה. למשל Thin Board Splines או Decision Trees. **יתרון**: ביטול גורם הטעות הנובע מבחירת h שאיננה קרובה ל f. **חיסרון**: בעיות חישוב ודרישה לכמות גדולה של דגימות (מרחב ההיפותזות גדול יותר). היתרונות והחסרונות הנ"ל הם ברמה התיאורטית ומשתמשים ב2 השיטות הנ"ל.

- יש המון שיטות למידה, אך אין שיטה אחת הטובה יותר מהאחרים לכל קבוצת אמון!

רגרסיה:

- רגרסיה ב 2 ממדים בעזרת Splines. מודל החיזוי בעזרת Thin Board Splines (יש ביכולתו להתאים את עצמו לדוגמאות שבאימון בצורה גמישה ואינה לינארית).

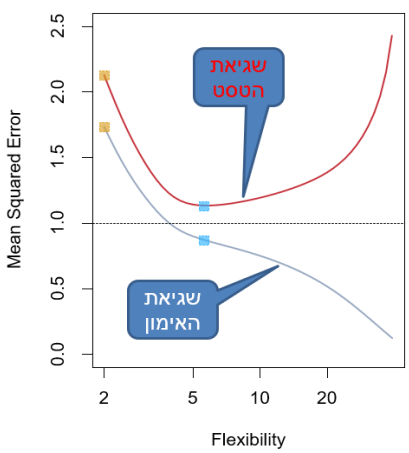
למעט היתרון בגמישות יתר במודל החיזוי, כך שהשגיאה קטנה עבור הדוגמאות הספציפיות על פיהן למד המודל, אלה החסרונות:

- גמישות יתר עשויה לגרום להתאמת יתר (שגיאת הכללה variance).
- מודל גמיש יהיה "כבד" יותר לחישוב.
- גמישות קשה לפרשנות (Interpretability).

- Residual Sum of Squares Errors

$$RSS = \sum_{x \in D} (t_x - h(x))^2$$
$$= \sum_{x \in D} (f(x) + \varepsilon - h(x))^2$$

- Mean Square Error= $MSE = \frac{1}{n} RSS$



שגיאות האימון קטנות עם הגמישות, שגיאות המבחן קטנות בהתחלה ואח"כ גדלות בהדרגה.

התופעה נקראת "התאמת יתר" **Over-fitting**.

השגיאה שיש למודל על נתוני המבחן מורכבת מ 3 סוגי שגיאה:
loss=Bias+Variance+irreducable

- על ה irreducible err אין שליטה, שאר השגיאות תלויות באלגוריתם הלמידה אך יש קשר ביניהם:
  - ככל שהמודל קשיח יותר (איננו גמיש) הוא עלול לא לקרב באופן הדוק את f: ייצר שגיאת Bias (התופעה נקראת **Underfitting**).
  - ככל שהמודל גמיש יותר, הוא עלול להתאים את עצמו לתבניות לא משמעותיות בנתוני האימון: וייצר שגיאת Variance – "התאמת יתר".

- שגיאת ה MSE היא התוחלת של ריבועי השגיאה:
על כל הדגימות D וכל הקלטים x לחלק למספר הדוגמאות (ממוצע):

$$MSE = E[(t - h_D(x))^2] =$$

$$\boxed{E[(h_D(x) - f(x))^2] + Var(\varepsilon)}$$

MSE = **ReducableError**+ var(**ε**)

ReducableError =**E**[h_D(x) −**f(x)**]^2 = Bias[h_D, **f**]^2 + Variance[h_D]

Bias[h_D, ***f***]= ***E***[h_D(x) −***f(x)***] => h – f => תוחלת ההפרשים

Variance[h_D] = ***E***[h_D(x) − ***E***[h_D(x)]^2] => h השונות של

- ברצוננו ללמוד f פונקציה בוליאנית מממד n כמה היפותזות יש במרחב החיפוש? 2^{2^n}.
- מרחב ההיפותזות מצטמצם אם רוצים שההיפותזות תהינה קונסיסטנטיות עם הדוגמאות. המרחב המצומצם נקרא version space.
- נתון אוסף נקודות {<x, t>} . נרצה למצוא היפותזה לינארית (קו ישר אם הממד הוא 1) הממזער את סכום ריבוע השגיאות. כלומר: נרצה (עבור ממד 1) למצוא שני פרמטרים: w0,w1 כך שהמשוואה. y=w1x+w0 תהיה משוואה של קו ישר המתאים ביותר לאוסף הנקודות.
  - MSE: $R^{n+1} \rightarrow R^+$
- MSE שייך למשפחה של פונקציות loss הממוזערות ע"י אלגוריתמי למידה.
- פונקציית הMSE ברגרסיה לינארית של משתנה בודד היא פרבולה דו-ממדית (קערה) עם נקודת מינימום אחת ויחידה. עבור 2 משקולות של היפותזה מחזירה את גודל השגיאה.
- כאשר יש n features, רגרסיה לינארית ניתנת להצגה: Y=WX כאשר W,X הם מערכים n+1 מממדים. בשיטת כתיבה זו: ערכו של x_0 הוא תמיד 1. ניתן אז להציג היפותזה באמצעות מכפלה סקלרית wx^T.
- לפעמים נוח לכתוב את הביאס בנפרד מווקטור המשקולות: $Y=h(x)=wx^T+b$.
- **NORMAL EQUATION:** נוסחה מפורשת (אנליטית) למשקולות ברגרסיה לינארית. נחשב נגזרת חלקית של פונקציית העלות, עבור כל אחד מ n המשקולות, מקבלים n+1 משוואות לינאריות שניתן לפתור באמצעות מציאת מטריצה הופכית.
- **חסרונות**: לא טוב ביצועית עבור ממדים גדולים כי הפיכת מטריצה של nxn בסיבוכיות $O(n^3)$, בנוסף לא חסכוני בזיכרון- ישנה דרישה שכל הנתונים יהיו בזיכרון, לעיתים רחוקות המטריצה לא הפיכה.
- **יתרונות**: ניתן לבצע בקלות בעזרת ספריות אלגברה פשוטות, מהירה בממדים קטנים, אין צורך בנרמול ובמציאת קצב למידה.
- X הינה מטריצה שבה השורות הם הדוגמאות. העמודות הם המאפיינים: עמודה ראשונה היא 1.
- בנקודת המינימום, n+1 הנגזרות החלקיות של פונקציית השגיאה מתאפסות:
- פונקציית השגיאה: ½ mean$(wx^T-t)^2$
- הגראדיינט מתאפס: $(wx^T-t)x=0$
- לאחר שינויים על 2 האגפים מתקבל שווקטור המשקולות האופטימליות: $\mathbf{w=(x^Tx)^{-1}x^Tt}$.
- שיטה נוספת למזעור שגיאות, מתאים גם למודלים לא לינאריים וגם לכאלה שמותאמים להרבה נתונים bigdata. **Gradient Descent on the loss Function:** (שמות שיטות המשתמשות בכך: GD/SGD/MiniBatch).

התחל מנקודה אקראית (של משקולות), בדוק על סמך השיפוע (גרדיאנט), לאיזה כיוון כדאי לרדת כך שתהיה ירידה הכי חזקה בעלות. חשב את הגראדיינט של פונקציית השגיאה בנקודה, ובצע צעד של

שינוי משקולות שיביא את הנקודה החדשה קרוב יותר לנקודת המינימום. גודל הצעד נקבע על פי פרמטר קצב הלמידה **λ**. עדכון סימולטני של כל המשקולות בסוף Epoc ייתן התכנסות Steepest descent.

- השינוי במשקולות הוא מינוס הנגזרת החלקית כפול קבוע קצת הלמידה **λ**.

מבצעים epocs עד שMSE קטן מספיק

או שמגיעים למספר איטרציות שנקבע

מראש ε.

$$\Delta w_i = -\lambda \frac{\partial loss_{D,h}(\mathrm{w})}{\partial \mathrm{w}_i}$$

$$= -\frac{2\lambda}{2m}\sum_{p\in D}(h_w(x^p) - t^p)\frac{\partial h_w(x^p)}{\partial \mathrm{w}_i}$$

- בכל epoc מעדכנים את הערך החדש של המשקולות עד שמגיעים לאופטימליות מספיקה כפי שנאמר לעיל. עדכון המשקולות נעשה על ידי: $W_i^{new} = W_i + \Delta W_i$.

- למידה איטית – כשהקערה צרה וארוכה. תופעה זו קורה: כאשר ל features יש התפלגות שונה וטווח ערכים שונה, למשל מספר החדרים: 2-6 וגודל הדירה: 50-200. מסקנה: רצוי לנרמל שדות.

$$= -\lambda \frac{\partial MSED(\mathrm{w})}{\partial \mathrm{w}_i}$$

$$= \frac{\lambda}{m}\sum_{p\in D}(t_p - y_p)\frac{\partial y_p}{\partial \mathrm{w}_i}$$

$$= \lambda\, mean(t-y)\nabla y$$

$$= \frac{\lambda}{m}\sum_{p\in D}(t_p - y_p)x_{ip}$$

- **יתרונות**: מהיר בממדים גדולים (אולי אפילו מיליון), ניתן לבצע גרסאות Online ו mini-batch שאינם דורשים זיכרון גדול (בממדים ענקיים) לכן מתאים יותר ל Big-data, GD כללי יותר: עובד גם עבור רגרסיה וגם עבור קלסיפיקציה, עובד גם עבור פונקציות עלות שונות (לאו דווקא סכום ריבועי השגיאה) וגם במקרים לא קמורים.

- **חסרונות**: דורש קביעת קצב למידה, התקדמות איטרטיבית יכולה להיות איטית, יש צורך בנרמול.

- **נרמול**: הסבה של ערכי השדה כך שיהיה לכולם אותו טווח (ולפעמים גם אותו ממוצע וסטית תקן) נרצה לנרמל את ה features (שדות) כך שערכם יהיה בסדר גודל דומה, למשל בין 0 ל 1 או בין [-1,1], עם ממוצע 0 ו/או עם סטית תקן 1. למשל: נניח שמספר החדרים: 2-8 (ממוצע 4) וגודל הדירה: 70-400 מ״ר וסטית תקן 1.5.

- **MinMax Scaling** יעביר ל [0,1]: (x – Min) / range. נרמול מספר החדרים. כיצד? $(room_i-2)/(8-2)$

נרמול גודל הדירה. כיצד? $(Size_i-70)/(400-70)$.

סוג נוסף של נרמול גורם לכך שלכל השדות יש ממוצע ערכים 0:

- **Mean-Normalization:** נמיר את ערכי השדות כך שיתנו ממוצע 0 (X-mean(X)).

- **SD-Scaling**: נחלק בסטיית התקן: (X/sd(X))
נקבל סטיית תקן 1 הערך החדש של הFeature משקף כמה המרחק שלו מהממוצע בסטיות תקן.

- **Standard-Normalization**: מחסר ממוצע ומחלק ב SD.

- מאפיינים כמותיים ואורדינליים: שכר, השכלה נוכל לנרמל: למשל כמספר ממשי (0,1) או (-1,1). ניתן לנרמל באופן סטנדרטי: התוחלת 0 וסטית התקן 1. מאפיינים איכותניים (קטגוריים) בינאריים נוכל לקודד 0/1 או 1/1-. מאפיין שיש לו k קטגוריות (למשל: ארץ מוצא): נוכל להפריד ל k מאפיינים בינאריים. יתכנו הרבה אופציות לקידוד.... והקידודים השונים עשויים להיות משמעותיים לגבי איכות המודל.

- 2 הרחבות המאפשרות שליטה והגדלה של הגמישות (יש מחיר variance): רגרסיה פולינומיאלית, רגרסיה לינארית לוקאלית ממושקלת: LWLR.

- לעיתים נזדקק להוסיף features חדשים שהם טרנספורמציות של המאפיינים הנתונים. כאשר ההיפותזה שלנו חלשה מידי (למשל לינארית או פשטנית מידי), נראה הרבה שגיאות בזמן האימון: Under fitting. טרנספורמציות על מאפיין בודד: חישוב גיל מתאריך הלידה, שטח מגרש מתוך אורך ורוחב. טרנספורמציות מורכבות: מכפלות של מאפיינים. כולל העלאה בריבוע, או מכפלות משולשות, חישוב מצב סוציו-אקונומי מהמיקוד, מההשכלה ומהגיל.

- בהוספת מכפלות של מאפיינים, נקבל **רגרסיה פולינומיאלית**. הרגרסיה נשארת לינארית גם אשר מתאימים פולינומים מדרגות גבוהות הקלט הגולמי נשאר כשהיה. אך את הרגרסיה הלינארית מבצעים במרחב Features שכולל מכפלות וחזקות של המאפיינים המקוריים.

- הערה פרקטית: חשוב מאוד לנרמל את הfeatures לפני שמעלים בחזקה מכיוון שהעלאה בחזקה גורמת לסדרי גודל שונים ("קערת" השגיאה איננה עגולה, ולכן, התכנסות איטית).

- אם ננסה להתאים פולינום מדרגה עצומה נקבל High Variance, Overfitting. הפולינום יעבור דרך כל נקודות הדגימה, שגיאת האימון תשאף ל0 אך תגדל עבור דגימות הטסט, עליהן הפולינום ישתולל.

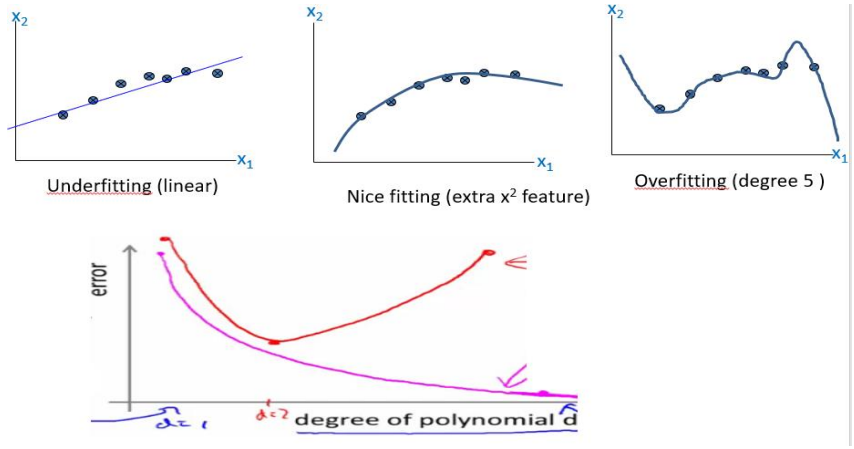
- בממד קלט גדול (בעל הרבה features) אם נוסיף את מכפלות המאפיינים, נקבל התפוצצות בממד (המון ויותר מדיי מאפיינים). הכמות הנוספת הינה קומבינציות של דרגת הפולינום מתוך מרחב הקלט הגולמי.

- **High-Variance** מרחב ההיפותזות גדול מידי ואין לנו מספיק data שתספק מספיק אילוצים כדי לבחור את ההיפותזה הטובה ביותר. מקבלים שגיאה שנובעת מהתאמת יתר לקבוצת אימון מסוימת.

- **High-Bias** שגיאה הנובעת מההיפותזה פשוטה מידי (למשל: לינארית).

- גם בקלסיפיקציה ניתן להוריד את שגיאות הBias ע"י טרנספורמציות ו/ או הוספת Features.

- יש **tradeoff** בין שגיאת הbias לשגיאת הvariance.



- **Locally weighted linear regression (LWR):** היבריד של שיטה לא פרמטרית ביחד עם רגרסיה לינארית לוקאלית. אימון: שומרים בזיכרון דוגמאות ומשתמשים בהם בכל פעם שזקוקים לחיזוי.

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

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

דוגמה לפונקציית דמיון בטא:

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

מרחק מחושב ע"פ הפונקציה הגאוסיאנית מסביב לנקודה: ככל ש ρ גדול יותר כך נלקחות בחשבון גם נקודות רחוקות.

ניתן לשלוט על הגמישות בעזרת פונקציית המשקל. 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. קיים דמיון רב ל KNN regression.

- כאשר יש כמויות נתונים ענקיות, בד"כ לא נשתמש ברגרסיה לינארית, אך אם נרצה נשתמש ב Stochastic GD.
- רגרסיה לינארית יתרונות: פשוט ומהיר לחישוב, מתאים כאשר ישנן מעט דוגמאות, ניתן לפרש בצורה פשטנית יחסית כיצד משפיע כל feature, ניתן להפחית את שגיאת הbias בעזרת הוספת features.
- חסרונות: High-Bias. בהרבה מקרים ההיפותזה פשטנית מדיי ולא ניתן להגמיש בגלל התפוצצות ממדים. Outliers: משנים רדיקלית את צורת ההיפותזה הנלמדת, למרות שהם מקריים. הנחות שונות על השגיאות ועל המאפיינים אינן מתקיימות תמיד במציאות:

- The inputs are correlated among themselves- not according to assumptions. (specially in time series).
- Non-constant variance in error terms.
- High-leverage points. Noise.
- Co-linearity. Relationship between features.

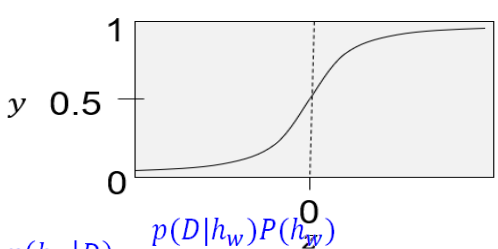
רגרסיה לוגיסטית\קלסיפיקציה בינארית:

מעוניינים לסווג, האם פרמטר שבודקים הוא 0 או 1 (לדוגמה: יש מחלה\אין מחלה). היפותזה עבור רגרסיה לוגיסטית היא מהצורה:

$$z = w_0 + \sum_i w_i x_i = wx$$

$$y = g(z) = \frac{1}{1 + e^{-z}}$$

g היא הפונקציה הלוגיסטית– Sigmoid.



אימון: מציאת משקולות של מישור מפריד. W שימקסם את $l_D(W)$.

חיזוי: חישוב ההסתברות של נקודה על פי מרחקה מהמישור המפריד.

- עבור קלסיפיקציה בינארית לא מומלץ להשתמש בפונקציית השגיאה של הMSE מכיוון שהפונקציה איננה קמורה (שלא כמו ברגרסיה לינארית) ולכן לא מובטח מינימום בודד, ומכיוון שקשה להצדיק תיאורטית MSE עבור קלסיפיקציה.

$$p(h_w|D) = \frac{p(D|h_w)P(h_w)}{P(D)}$$

- בהנחת **אי תלות** בין הדוגמאות: ההסתברות שמודל W חוזה נכון את כל הדוגמאות בD שווה למכפלת ההסתברות שכל דוגמה נחזית נכון. משתמשים בחוק bayes.

$$l_D(w) = \prod_{p:t_p=1} y_p \prod_{p:t_p=0} (1 - y_p)$$

$$y = h_w(x_p)$$

- מכיוון שרוצים למקסם את ההסתברות $p(h \mid D)$, ניתן להתעלם מההסתברות ל- $p(D)$ D הקבוע שאינו תלוי בw. נתעלם גם מ$p(w)$ (הפריור של ההיפותזה) : מחוסר ידע מניחים שלכל ההיפותזות יש הסתברות שווה.

- נרצה למקסם את ה Likelihood שההיפותזה מסבירה את D. הliklihood של h_w ביחס לD.
אידיאלית h_w צריך לתת הסתברות 1 לדוגמאות חיוביות ו0 לדוגמאות שליליות ואז ככל ש h_w אינו מדויק בחיזוי כך $l_D(W) < 1$.

- ממוצע גיאומטרי לפי m דוגמאות ייתן את ההסתברות הממוצעת לחזות נכון בדוגמא x בעזרת w.

- בגלל מונוטוניות של פונקציית הlog נמקסם את log של הliklihood ובכך נשתמש בתכונות הlog כדי להגיע לתוצאה :

$$loglikelihood_D(w) = \sum_p t_p \log(y) + \sum_p (1 - t_p) \log(1 - y)$$

- אם ניקח ממוצע (של הדוגמאות ב D)
נקבל את ה Log של ההסתברות
הממוצעת לחיזוי נכון של הדוגמאות
ב D. נהפוך את הסימן כדי לקבל מספרים חיוביים - שאותם נרצה למזער, נקבל את נוסחת השגיאה אותה נרצה למזער :

$$CrossEntropy(y,t) = -\frac{1}{m}\left(\sum_p t_p \log(y) + \sum_p (1 - t_p) \log(1 - y)\right)$$

- ממוצע רגיל של
הלוגים אקוויוולנטי ללוג הממוצע הגיאומטרי של ההסתברויות.

- וזו דרך כללית להתאים היפותזות y למטרת קלסיפיקציה. עבור רגרסיה לוגיסטית :
$y=h_w(x)=g(wx+b)$.

- נשתמש במינוס ה cross-entropy לחישוב עלות השגיאה פר דוגמה, עבור דוגמא בודדת <x, t> כאשר ההיפותזה היא $y=g(wx+b)$:

- שגיאת ה Cross-entropy עבור קבוצת אימון

$$C(y,t) = -t\ln(y) - (1-t)\ln(1-y)$$

הכוללת m דוגמאות :

$$Loss = CE_D(w) = \frac{1}{m}\sum_{p \in D} C(y_p, t_p)$$

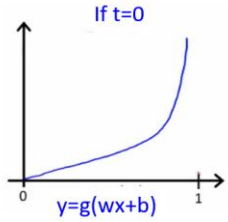
- לדוגמה : בקובצת האימון 2 דוגמאות : תמונה של תפוז ותמונה של לימון. נתון שהקלסיפייר סיווג את תמונת התפוז בהסתברות של 0.8 ואת תמונת הלימון בהסתברות של 0.7. השגיאה :

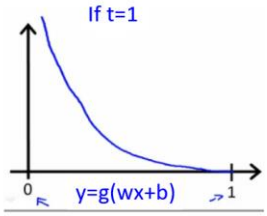
על תמונת התפוז : $-1(\ln(0.8))=0.22$

על תמונת הלימון : $-(1-0)\ln(1-0.7))=1.2$

השגיאה על כל קבוצת האימון (ממוצע) :
$(0.22+1.2)/2=0.71$

השערוך של ההסתברות הממוצעת לחיזוי נכון של הקלסיפייר : $e^{0.71} = 0.49$





השינוי במשקולות עבור GD או mini-batch על קבוצה של m דוגמאות:
ממוצע של השינויים:

$$\Delta w_i = \lambda/m \sum_{p \in D} (t_p - y_p) x_i$$

- **<u>נרמול:</u>** גם ברגרסיה לוגיסטית ה CE היא קערה

  בעלת מינימום אחד. רצוי מאוד לנרמל כדי שהקערה

  לא תהיה צרה וארוכה.

- כיצד נשתמש במודלי קלסיפיקציה בינארית עבור Mult-Class? אחת הגישות: One vs Rest: נבנה קלסיפייר לכל קטגוריה: אימון: נלמד להפריד קטגוריה אחת מכל השאר (כך לכל הקטגוריות).
  חיזוי: בהינתן דוגמה לסיווג, נבדוק חזית של כל הקלסיפיירים. נחזיר את הקטגוריה שהקלסיפייר שלה חוזה את ההסתברות הכי גבוהה.

- גם בקלסיפיקציה כאשר נרצה לעבור לייצוג פולינומי עבור ההיפותזה האופטימלית כשמרחב הקלט גדול נקבל התפוצצות ממדים. דוגמה מזערית (לשם הדוגמה): פולינום מדרגה 3: 9 קלטים

  $$x_1, x_2, x_1x_2, x_1^2, x_2^2, x_1^2x_2, x_1x_2^2, x_1^3, x_2^3$$

  במקרה כזה שגיאת הVariance תעלה והמודל יבצע Overfitting לדוגמאות האימון. להתפוצצות ממדים ישנה עלות חישובית גבוהה, קשה כך קשה להסביר את המודל ונדרשים הרבה נתונים כדי להימנע משגיאת Variance גבוהה.

- כאשר המודל אינו מספיק גמיש ושגיאת Bias גבוהה:
  - משלמים למהנדסים "יצירתיים" כדי שיפיקו טרנספורמציות למאפיינים "טובים".
  - משתמשים במודלים גמישים ובטכניקות לביצוע טרנספורמציות אוטומטיות (רשתות נוירונים, Kernels).

- כאשר ממד הקלט גבוה או שהוספנו יותר מידי Features- שגיאת הוואריאנס הופכת לגבוהה וזמני העיבוד מתקלקלים:
  - השקעה כספית: קונים חומרה חזקה, משלמים כדי לקבל יותר נתונים מסווגים.
  - משתמשים בטכניקות לצמצום מספר הממדים (למשל ב PCA).
  - משתמשים בטכניקות לצמצום שגיאת ה variance:
    - מורידים גמישות המודל
    - משתמשים בטכניקות לרגולריזציה
    - מוסיפים (או מחוללים) נתונים

- הערכת הצלחת חיזוי של קלסיפיקציה בד"כ נעשה על ידי confusion matrix (מטריצת העמימות) שמאפשרת למדוד זאת, ולא על ידי Cross Entropy או MSE ברגרסיה, כיוון שפחות אינטואיטיבית לבני אדם. KPIs והמטריצה:

- עבור המשתמשים (מהעולם העסקי) עדיף למדוד ביצועים בדרכים אחרות ויותר מובנות: סיווגים שגויים (Missclassifications), Precision של הקלסיפייר: כמה לסמוך על אבחנה חיובית, כמה אזעקות שווא. "אחזור" Recall ? כמה לסמוך על אבחנה שלילית, כמה "פספוסים" (חולים שלא אובחנו), ועוד דרכים.

- **True/False**- האם החיזוי צדק\לא צדק. **Positive/Negative**- קרו\לא קרו לפי החיזוי **ולא** בפועל.

True Positive- כמה קרו לפי החיזוי (positives) **וגם** החיזוי צדק (החיזוי צדק= true, חולים בפועל **וגם** נחזו שחולים).

False Positive- כמה קרו לפי החיזוי **אך** החיזוי טעה (החיזוי טעה- false, בריאים בפועל- **אך** נחזו כחולים).

False Negative- כמה **לא** קרו לפי החיזוי **וגם** החיזוי טעה (החיזוי טעה- false, חולים בפועל שסווגו כבריאים).

True Negative- כמה **לא** קרו בפועל וגם החיזוי צדק (החיזוי צדק= true, כמה בריאים בפועל שסווגו כבריאים).

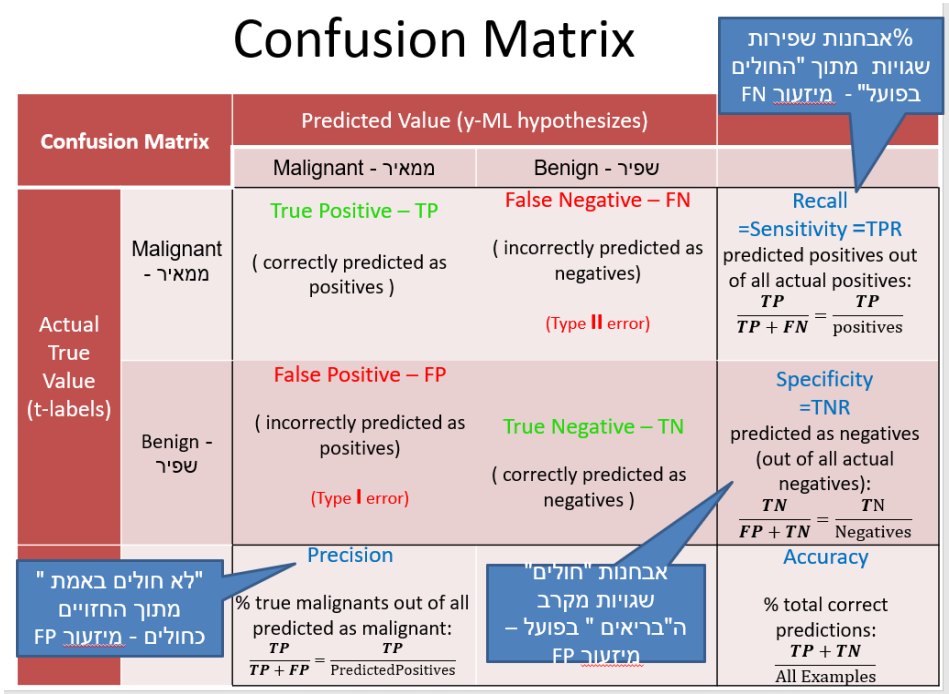
מדדים שונים על פי הטבלה להצלחה של חיזוי של קלסיפיקציה :

- **Recall**=Sensitivity=TPR- מדד למדידת כמה נחזו שקרו מתוך הכמות הכוללת של דוגמאות ש**בפועל** קרו (כמה נחזו שחולים, מתוך כמות החולים הכוללת בפועל). כלומר הסיכוי לסווג נכון מקרה חיובי מתוך כלל החיוביים בפועל.

- Specificity=TNR- מדד למדידת כמה נחזו ש**לא** קרו מתוך הכמות הכוללת של דוגמאות ש**בפועל לא** קרו (כמה נחזו כבריאים, מתוך כמות הבריאים הכוללת בפועל). כלומר הסיכוי לנחש נכון מקרה שלילי מתוך כלל השליליים בפועל.

- Precision- מדד למדידת כמה קרו **בפועל** מתוך הכמות של אלה ש**נחזו שלא** קרו (כמות החולים **בפועל**, מתוך כמות אלה ש**נחזו** כחולים). כלומר הסיכוי לסווג נכון מתוך כלל המסווגים חיובית על פי החיזוי.

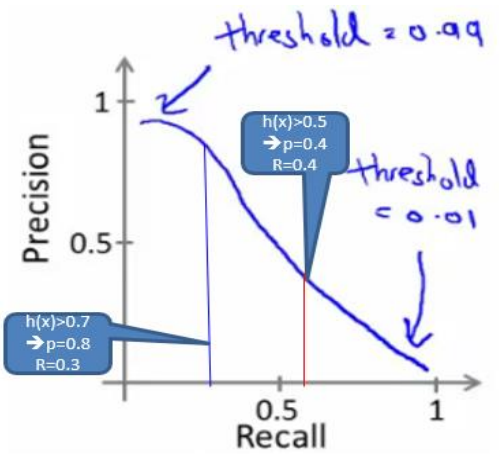
- Accuracy- מדד למדידת כמה **נחזו** בצורה מוצלחת מתוך הכמות הכוללת של הדוגמאות (כמות הדוגמאות שסווגו נכון, צדקו=true בכולן אל מול ערך הlabel, מתוך הכמות הכוללת של דוגמאות שנבחנו).



- במקרים לא מאוזנים מעדיפים Precision על פני TNR=Specificity כדי למדוד FP כיוון שקבוצת השלילים במקרים כאלו יכולה להיות הרבה יותר גדולה מאלו שמסווגים חיובי. *בשני המדדים רוצים למזער FP.

- בקלסיפיקציה מחשבים הסתברות, אך ההחלטה כיצד לסווג תלויה בסף (Threshold).
למשל: בד"כ מחליטים לסווג לקטגוריה, אם הסיכוי גדול מ 0.5. נוכל לשנות את הסף וכך לשנות את הפרופורציות של Precision, Recall בהתאם להשפעתם הכלכלית. נוכל לצייר גרף שמראה כיצד משתנים הדיוק והאחזור כפונקציה של הסף אם נשתמש בסף $h_w(x)>0.5$, נקבל מדדי דיוק ואחזור מסוימים. יש **tradeoff** בין שני המדדים: כשהסף נמוך ה- recall משתפר והדיוק יורד, ולהיפך.
למשל: אם נרצה להגדיל את רמת הביטחון שלנו (כדי לא לשגות כאשר מבשרים על סרטן), נגביה את הסף: $h_w(x)>0.7$. נקבל אז, דיוק גבוה יותר, אך ירידה באחזור (חלק ממקרי הסרטן הגבוליים, לא יסווגו כסרטן).

- <u>Skewed Analysis</u>- במקרים בהם אחת הקטגוריות נדירה, מדידת ה- Accuracy בלבד לא תעזור: אנחנו מעוניינים ש2 המדדים P, R יהיו גבוהים. לדוגמה, כשהסיכוי ה- prior למחלה באוכלוסייה הוא זניח, לדוגמה 0.005, הקלסיפייר המודד לפי accuracy=0.005 יחזה בצורה "טיפשית" על כל אדם "חולה".



- **<u>F-Measure=F-Score</u>**- שילוב בין recall ל- precision על ידי ממוצע הרמוני ביניהם. נהוג להשתמש בו על מנת להשוות הצלחה בין אלגוריתמי למידה שונים. ניתן למשקל את ה- recall וה- precision במקרים שונים אם רוצים להעדיף אחד על פני השני:

$$F-Measure = \frac{1}{\alpha \cdot \frac{1}{Precision} + (1-\alpha) \cdot \frac{1}{Recall}}$$

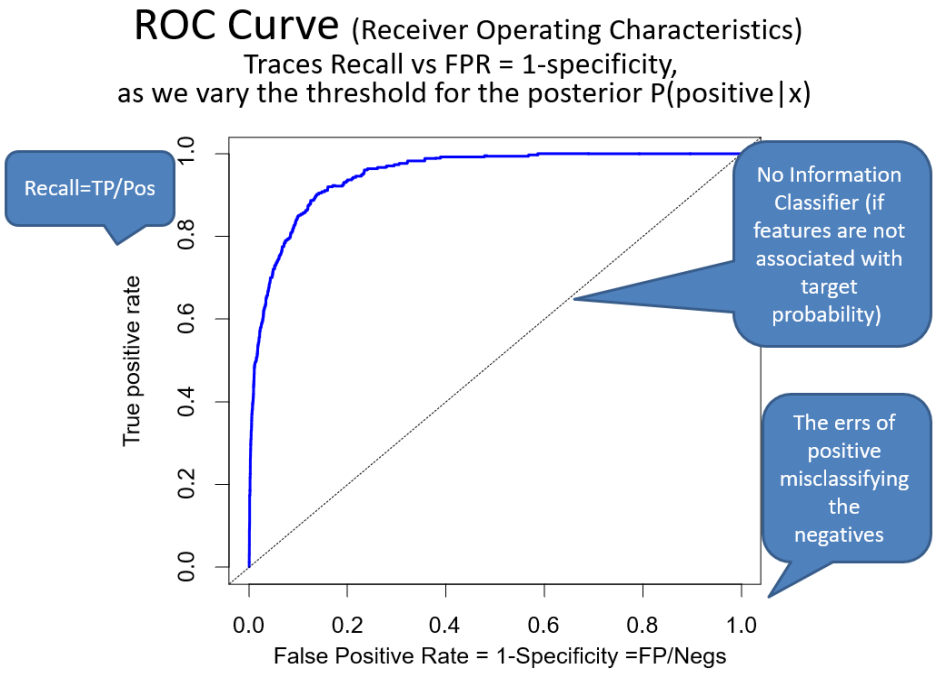
<u>אלפא גדול מחצי</u>: מתן עדיפות ל- precision.

<u>אלפא קטן מחצי</u>: מתן עדיפות ל- recall.

<u>אלפא שווה חצי</u>: נוסחה בה P ו- R נלקחים בחשבון בצורה שווה:

$$F1 = 2 \cdot \frac{P \cdot R}{P + R}$$

- **<u>מדד ROC:</u>** על פיו מחושב ה- **AUC**



- בקלסיפיירים טובים העקומה מוטית לצד שמאל למעלה. ניתן למדוד את "טיב" הקלסיפייר באמצעות **AUC**-Area Under the Curve. ה- AUC המקסימלי הוא 1 . עבור קלסיפייר שמגריל רנדום, נקבל AUC=0.5 (קו מרוסק) כאשר מופעל קבוצת טסט.

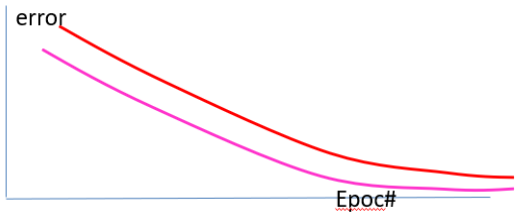
כלים לקביעת היפר-פרמטרים (קובעים דברים כמו : מתי לעצור את האימון, כמה איטרציות לבצע, באיזה קצב למידה להשתמש, באיזו דרגת פולינום להשתמש, ועוד...) על מנת לדעת את הביצועים הצפויים מהמודל שיצרנו לכשירות על טסט אמיתי :

- השיטה הפשוטה והמקובלת : להקצות חלק מקבוצת האימון לוולידציה.

- **פרקטיקה מתי לעצור ב- GD:** פופולרי : שימוש בוולידציה עבור קריטריון עצירה : כאשר שגיאת הוולידציה נמוכה ואינה משתפרת ע"י עוד אימונים, אך כאשר קבוצת הוולידציה קטנה מידי לא נוכל להשתמש בה לעצירה.

- **פרקטיקה: כיצד לבחור קצב למידה LR עבור GD:** אם רואים כי השגיאה עולה בהתמדה או לא יורדת, יתכן והקצב גבוה מידי והעדכון זורק לכיוון השני של המינימום לנקודה גבוהה יותר, לכן כדאי לנסות להוריד את הקצב. מצד שני אם הקצב נמוך מדיי תהיה התקדמות איטית מדיי. **הפתרון:** בדיקת מספר ערכים שונים עבור ההיפר-פרמטרים על שגיאת הוולידציה, לדוגמה ב- GD : נבדוק מול קבוצת וולידציה של מספר קצבי למידה 0.001,0.05,0.01... נבחר ב- LR הגדול ביותר שאינו מרע משמעותית את שגיאת הוולידציה.

- ניתן לאבחן מצבי התאמת יתר\חסר על ידי גרף עקומת הלמידה של שגיאת הוולידציה אל מול שגיאת האימון. כך נראה הגרף באופן נורמלי כך ששגיאת הוולידציה טיפה מעל שגיאת האימון. הצלחה- 2 השגיאות יורדות עד לשגיאה שמספקת אותנו.



- שימוש בוולידציה כדי לשפר את מנגנון הלמידה:
  - ננסה מודלים/אלגוריתמי למידה שונים ננסה קצבי למידה שונים, תנאי עצירה שונים.
  - ננסה לחבר מודלים שונים (ensemble).
  - נאתר מודל ופרמטרים שממזערים את השגיאה על הוולידציה.
- <u>Under-estimation</u>- כאשר שגיאת הוולידציה קטנה משגיאת הטסט כנראה בגלל Overfitting למודל הוולידציה.
- <u>Over-estimation</u>- כאשר שגיאת הוולידציה גדולה מאשר שגיאת הטסט כנראה בגלל חוסר בנתונים עבור הטסט.

<u>סיבות לכך שהקצאת חלק מקבוצת האימון לוולידציה היא אינה שיטה טובה</u>:

- אם הוולידציה משמעותית, קבוצת האימון תהיה קטנה יותר: ככל שמבחן הוולידציה גדול יותר, יש פחות דוגמאות אימון והמודל יהיה פחות טוב ממודל שיואמן על כל הדוגמאות. לכן, נקבל "הערכת יתר" של השגיאה.
- בחירה רנדומלית של ווליידציה עלולה לא לשקף את המבחן האמיתי.
  - הוריאנס של כל קבוצות הוולידציה האפשריות עלול להיות גדול: מדגם ווליידציה קטן מידי עלול לא לשקף את הטסט.
  - התאמת יתר: בחירת ההיפר-פרמטרים עלולה לגרום למודל להיות מותאם מידי לוולידציה.
    שיערוך השגיאה הצפויה עלול להיות מוטה לקבוצת הוולידציה (הערכת חסר).

<u>Cross Validation</u>: שיטה כללית למזעור השגיאה על הוולידציה (ועל האימון יחד) כמה שאפשר. חלוקה של קבוצת האימון להרבה קבוצות שונות של זוגות (אימון, וולידציה), אימון כל מודל בנפרד וחישוב שגיאתו על הוולידציה (לכל זוג) ולבסוף מיצוע כל שגיאות הוולידציה שנמצאו על ידי (MSE, CE, F, P, R). CV **כבד** ויקר חישובית אך משערך טוב יותר את שגיאת הטסט ולעיתים קרובות גם מאפשר חיזוי טוב יותר (ע"י שימוש באנסמבל). לדוגמה עבור MSE:

$$CVmse = \frac{1}{n}\sum_{v} mse_v$$

- עבור חיזוי בשלב הטסט/פרודקשן:
  - ניתן לבנות את המודל הסופי מקבוצת האימון כולה (נהמר שהערכת השגיאה שעשינו רק על חלק מהנתונים תקיפה גם למודל שאומן על יותר נתונים).
  - לחילופין, נשתמש **באנסמבל**: נבחר ב- k המודלים עם שגיאת ווליידציה הנמוכה ביותר ונמצע את התוצאות (לרגרסיה) או נסווג על פי החלטת הרוב (קלסיפיקציה). **יתרון**: שימוש במודלים גדולים עבור אנסמבל מפחית את שגיאת הוריאנס.

מימושים:

- **<u>K-fold Cross Validation</u>:** חלק את ה הנתונים המתויגים ל- K קבוצות: שמור K/1 מהנתונים כקבוצת ולידיה: אמן על K/(K-1) מהנתונים ובדוק על הוולידציה. ביצוע של K אימונים שונים בכל פעם על קבוצת ווליידציה אחרת כדי לקבל K תוצאות שונות ל- validation loss. לבסוף, מיצוע התוצאות כדי לקבל הערכה לשגיאת הטסט או בחירה במודל(ים) עם הביצועים הטובים ביותר או צירופם ל- Ensemble ומיצועם. **למשל**, נחלק את ה- training ל- 10 קבוצות: נאמן על 9 ונשתמש בעשירית כ- validation. נמשיך כך 10 פעמים, כשבכל פעם בוחרים קבוצת validation אחרת.

- **<u>Leave One Out Cross Validation (LOOCV):</u>** ביצוע דומה למהלך של K-fold, אך בשונה: בחירה של דוגמה **אחת** לוולידציה במקום K\1, וכל שאר הדוגמאות (m-1) עבור אימון של מודל. ביצור ההליך ליצירת m מודלים שונים. שיערוך השגיאה ע"י מיצוע של השגיאות על כל הדוגמאות ב-D. בד"כ LOOCV משערך שגיאה טוב מאוד, אך יקר מאוד לחישוב (ברגרסיה לינארית יש שיטה יעילה מאוד לעשות LOOCV, במחיר של רגרסיה רגילה).

- נתונים מספריים עבור כמה מודלים\חלוקות אפשריים יש:
  - כל החלוקות השונות ל- Train_validation (מודלים): $O(2^m)$.
  - Leave-P-Out CV: כל החלוקות האפשריות לשימוש ב- v דוגמאות בתור וולידציה: C^m_v.
  - Leave-One-Out CV: m.
  - K-fold CV: k.

- **<u>The Boot Strap Method:</u>** יצירת הרבה קבוצות אימון מקבוצת אימון אחת שאותה דוגמים רנדומית עם חזרות:
  - בכל פעם שיוצרים מדגם אימון של m דוגמאות מתוך m דוגמאות מקור, חלק מדוגמאות לא מופיעות במדגם והן נלקחות כוולידציה.
  - ניתן לייצר **למשל** 1000 מדגמים של 100 דוגמאות מתוך 100 דוגמאות מקוריות. ממצעים את תוצאות הוולידציה על 1000 המדגמים שיצרנו

מסתבר שיכולות השערוך משתפרות ככל שניצור עוד מדגמים מבלי להוסיף שום דוגמה חדשה.

- שיטות להפחתת שגיאת הוריאנס באופן כללי: Feature Selection, Regularization, Ensembles, הוספת דאטה, עצירה מוקדמת.

- <u>Feature Selection</u>: אלגוריתמים לבחירת תת קבוצה של פיצ'רים **חשובים** מתוך הקבוצה הנתונה. הפרוצדורה האולטימטיבית:

- לכל

Somewhat more efficient heuristic

- For k=1 to n
  - Train all $\binom{n}{k}$ models generated by A with k out of n features
    Select the best model (minimal training loss)
    set M_k
- Select the best model among $\{M_i\}$ with minimal CV loss

בוחרים את המודל הטוב ביותר עבור k מאפיינים רק בעזרת שגיאת האימון

CV מתבצע לכל k

בחירה של V (קבוצת features), בנה מודל M_v ובדוק את שיערוך השגיאה (CV) שלו.

בחר בV בעלת שיערוך השגיאה המינימלית.

האלגוריתמים של Feature Selection פחות יעילים, אפילו תחת היוריסטיקות מסוימות. 3 גישות היוריסטיות: Forward Selection, Backward Selection, Mix (Hybrid Selection).

- **<u>Forward Selection (Greedy):</u>** התחל ממודל ללא features. בכל שלב k עבור על כל feature חסר, ובדוק את שגיאת האימון המתקבלת ע"י הוספתו. בחר את ה- feature שמפחית הכי הרבה את שגיאת האימון וצור מודל M_k הכולל את ה- feature. לכל k, בצע הערכת שגיאה באמצעות CV למודל הכי טוב, ובחר את ה- k הטוב ביותר.

  1. Begin with null model (zero features) M_0: $y=w_0$
     train and save its loss score
  2. K=1….n, until (stop criteria; e.g. loss is small enough)
     For all features F_k not in the model k-1:
     - Try adding each of the missing features, train and save loss
     - Create M_k by adding the feature that provides the best loss
  3. Choose the best M_k using CV

- **<u>Backward Selection:</u>** התחל ממודל M_n הכולל את כל ה- features. בכל שלב k ועבור כל feature במודל, בדוק את שגיאת האימון המתקבלת ע"י הסרתו. הסר את ה- feature שגורם לשגיאה הקטנה ביותר וצור מודל M_k ללא אותו פיצ'ר. בצע הערכת שגיאה באמצעות CV למודל M_k הכי טוב בעל k=1….n פיצ'רים, ובחר את ה- k הטוב ביותר.

  1. Begin with a full model
  2. K=n-1…1, until (stop criteria)
     Fit all models that contain all k features but one.
     Remove the feature with the model with least MSE impact
  3. Choose the best M_k using CV

- **<u>Hybrid:</u>**

  1. Begin with Null model
  2. K=1…n, until (stop criteria)
     Add a feature as in Forward selection
     Remove a feature that no longer provide improvement
  3. Choose the best M_k using CV

<u>רגולריזציה:</u> המטרה היא להוריד שגיאת variance. מקרה פרטי של feature selection. הענשת המשקולות של נוסחת השגיאה שממזערים (מקדמי פולינום קטנים אינם מאפשרים "פיתולים עזים" ולכן מונעים התאמת יתר).

- ע"י הענשת משקולות, נוכל "לבקש" ממנגנון הלמידה "להשתדל" לא להקצות מקדמים גדולים מדיי. **נקבל מודל חיזוי חלק יותר**.
- איננו יודעים על איזו משקולת לוותר ולכן נעניש את כולן.

- לא נהוג לא להעניש את הביאס w_0 מכיוון שיש לו תפקיד עקרוני ברגרסיה:
הוא הערך החזוי כאשר הקלטים שווים ל- 0 והוא ממוצע התחזיות כאשר כל הקלטים מנורמלים סביב ממוצע 0.

- **γ** קבוע הרגולריזציה. פשרה בין 2 מטרות, פשטות מול כמות השגיאות: אם נבחר **γ** גדול מידיי (למשל 10^{10}), נגרום ל- underfitting, ז.א. היפותזה שמתקרבת ל- $y=w_0$.

- **אינטואיציה:** חיסור נתח של **γ** מתוך השגיאה שחישבנו (MSE וכיוצא בזה).

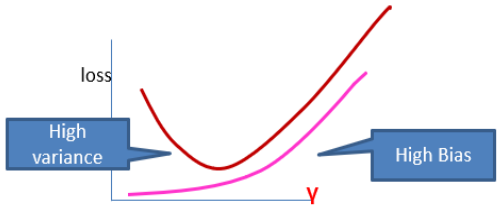
סוגי רגולריזציה:

Ridge regularization: מתבצעת על ידי חיסור מכפלה של **γ** בנורמה 2 של כל משקולת (סכום הנורמות במרחב L2- מרחק מראשית הצירים בריבוע, כפול קבוע הרגולריזציה **γ**):

$$regloss(w)=loss(w)+\frac{\gamma}{2}\sum_{j=1}^{n} w_j^2)$$

- עקומת השגיאה על האימון ועל הוולידציה כפונקציה של **γ**:

- נהוג לעשות למאפיינים Scale normalization ע"י חלוקה בסטיית תקן. סטיית התקן לאחר הנרמול היא 1 לכל המאפיינים שנורמלו. כאשר סטיית התקן זהה בכל המאפיינים, גם המשקולות תהינה באותו סדר גודל והעונש γ ישפיע באותה מידה, לדוגמה: המשקולות של גיל בשנים ומשקל בגרמים הם בסדרי גודל שונים. לא הגיוני להעניש אותם במידה שווה.



- כלל עדכון המשקולות, לדוגמה ב- GD:

$$w = w - \lambda\nabla loss - \gamma w$$

- יתרונות מול Feature Selection:
  - יעילות חישובית.
  - Features בעלי תרומה קטנה, לא יסולקו בהכרח אלא ישתתפו בחיזוי (משקולות קטנים).
  - עמידות טובה יותר לרעשים: רעש בהרבה features מתמצע וכן מופחת.
  - ניתן להרחבה לכל אלגוריתם למידה פרמטרי (למשל רשתות נוירונים).

- חסרון: ב L2, נשארים הרבה features, קשה להסביר מודל מורכב.

Lasso Regularization: באופן דומה ל- Ridge, אך החיסור משתמש בנורמה במרחב L1, כלומר בערך המוחלט של המשקות.

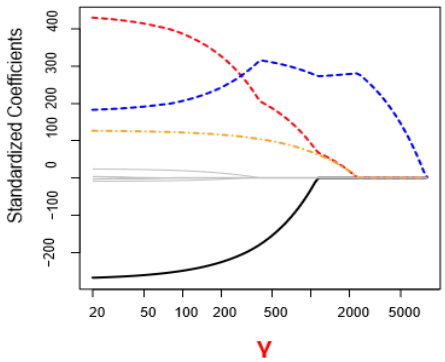
- L1 מסוגל לבצע Feature Selection: ככל שנגדיל את ה"עונש" משתנים ייעלמו (יקבלו משקל 0). ב- Ridge המשתנים יקבלו משקולות נמוכים אבל לא ייעלמו לגמרי.

- כלל עדכון המשקולות, לדוגמה ב- GD:

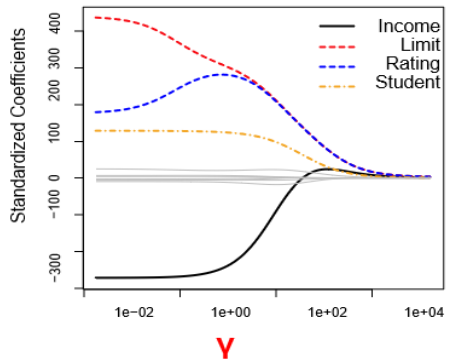
$$\text{Regloss}_{L1}= loss_D(w)+\frac{\gamma}{n}\sum_{j=1}^{n}|w_j|$$

$$w = w - \lambda\nabla loss - \gamma(sign(w))$$

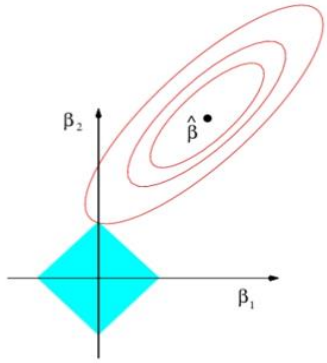
- גם ברגולריזציית לאסו: ככל שנגדיל את גמא (העונש) כך הביאס יעלה והווריאנס ירד.



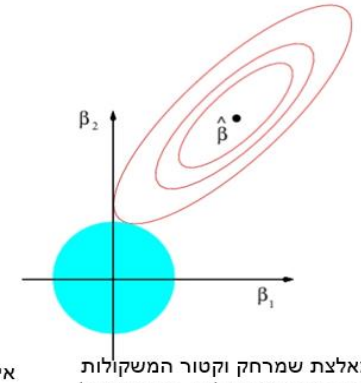
לאסו L1: ככל שגמא גדל, משתנים נעלמים



ב- L2 Ridge המשקולות דועכות אך המשתנים לא נעלמים.



אילוץ L1 יוצר "יהלום": פינות חדות על הצירים, מפגש היהלום עם אליפסת ה- MSE מקבל מינימום לרוב על הצירים.



נורמה L2 מאלצת שמרחק וקטור המשקולות מ (0,0) יהיה ברדיוס מסוים (אין פינות חדות). לכן אליפסת ה- MSE תיחתך עם האילוץ לאו דווקא על הצירים.

- כשהמשקולות גבוהים, ridge מנחיתה אותם לכיוון 0 ריבועית. כשהמשקולות נמוכים, Lasso מנחיתה אותם חזק יותר מ- ridge.

- ברגולריזציית לאסו אספקט הפרשנות טוב יותר מאשר רגולריזציית רידג' (ניתן להסבר), אך בד"כ רידג' ייתן תוצאות טובות יותר מכיוון שהוא ממצע משתנים קורלטיביים ובכך מסייע בהפחתת הרעש. בד"כ כאשר ישנם מעט פיצ'רים חשובים, והרבה מהם מכניסים רעש ולא משפיעים הרבה על החיזוי, Lasso יהיה טוב יותר. **בעזרת CV ניתן להחליט באיזו נורמה כדאי להשתמש ומהו ערכו של קבוע הרגולריזצייה**.

- **<u>Elastic Net: 0.5Lasso+0.5Ridge</u>**: במשקולות הקטנים: הלאסו משפיע ובמשקולות הגבוהים, הרידג' (אבל הלאסו ממתן קצת).

גישות נוספות להפחתת Overfitting:

<u>Ensembles - חכמת ההמון, צירוף של הרבה מודלים שונים:</u>

- **Bagging:** מאמנים מודלים שונים על תת קבוצות שונות של ה- data. למשל תוך שימוש ב- N-fold cross validation. את הפלטים ממצעים (או מבצעים סיווג- לקיחת המודל\ים הטוב\ים ביותר על פי הצבעת הרוב).

- **BootStrap:** מייצרים הרבה קבוצות אימון שונות ע"י דגימה אקראית עם **חזרות** מתוך קבוצת אימון יחידה. לכל קבוצה מותאמת קבוצת ולידציה הכוללת דוגמאות שלא נלקחו לאותה דגימה.

- **Boosting:** מייצרים קבוצה של מודלים **חלשים**, כל מודל לומד מהטעויות שהמודלים האחרים לא הצליחו לסווג, ומסווג את המקרים בהתאם. משתמשים באותו DATA בכל המודלים, אך מגדילים

את משקלן (תדירות הופעתן) של דוגמאות שלא סווגו נכון על ידי מודלים קודמים. את המודלים השונים שהתקבלו ניתן למצע\הצבעת הרוב. **חסרון:** רגיש מאד ל- outliers ועלול לגרום לווריאנס גבוה במידה שהמודלים שהתקבלו אינם חלשים כל כך.

אלגוריתם למידה נוסף:

KNN – K nearest neighbors: בהינתן אוסף דוגמאות אימון ודוגמת מבחן חדשה, מחשבים את ההסתברות האפוסטריורית של כל הקטגוריות, כלומר ההסתברות לקטגוריה כלשהי בהינתן דוגמת המבחן. הדבר נעשה על פי הצבעת K השכנים הקרובים ביותר לדוגמת המבחן. נמצא את הקבוצה N_x המכילה את K הדוגמאות מתוך D, ה"קרובות" ביותר ל x: ההסתברות לקטגוריה j מחושבת על פי מספר הדוגמאות מ N_x שסווגו לקטגוריה j:

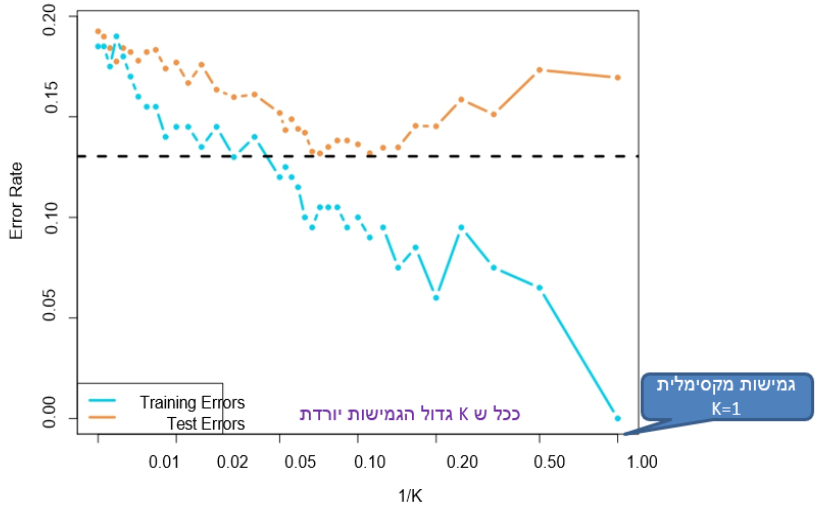
$$p(y=j|x,D)=\frac{1}{K}\sum_{i\in N_x} I(y_i=j)$$

בוחרים את הקטגוריה בעלת ההסתברות המקסימלית.

התחזית עבור הדוגמה x היא לכן על פי "הצבעת" הרוב מתוך N_x.

ב- 1NN מקבלים גבולות החלטה מסובכים וגמישים להפליא. ככל שמגדילים את k מאבדים גמישות (מגדילים שגיאת ביאס אבל מפחיתים שגיאת ווריאנס). כאשר k=n מקבלים שההסתברות לקטגוריה j היא השערוך של הפריור של הקטגוריה: P(y=j).

השוואת שגיאת האימון ושגיאת המבחן כפונקציה של הגמישות 1/k



- אלגוריתם הקלסיפיקציה (Voting):

בהינתן שאילתה x_q, עבור על k הדוגמאות הכי קרובות: החזר את הקטגוריה הכי תדירה מתוך k השכנים. ההסתברות לקטגוריה תשוערך על פי מספר השכנים השייכים לקטגוריה חלקי k.

- אלגוריתם הרגרסיה (Mean):

בהינתן שאילתה x_q, עבור על k הדוגמאות הכי קרובות: החזר את ממוצע הערכים על פני k הדוגמאות השכנות.

- הקשר לאנסמבל: כל דוגמא היא מייצרת מודל קלסיפייר עם שגיאת ווריאנס גדולה. הרבה נקודות – חכמת ההמון: שגיאת הווריאנס יורדת.

- VDM מרחק בין זוג ערכים פרופורציוני להפרש ביכולות החיזוי של הערך (סכום על כל הסיווגים).
ככל שערכם משפיעים בצורה שונה על הסיווג, כך המרחק ביניהם יהיה גדול יותר. כאשר n=2 :
המרחק בין 2 ערכים הוא סכום ריבוע ההפרשים של ההסתברויות המחלקות. ז.א. 2 ערכים שונים זה מזה אם יש הבדל בהשפעתם על הסתברות המחלקות.

– Value difference measure (VDM):

$$\delta(val_i, val_j) = \sum_{h=1}^{\#\text{classes}} |P(c_h|val_i) - P(c_h|val_j)|^n$$

ניתן לשערך ע"י ספירת הדוגמאות המסווגות כ c_h, מתוך אלו בעלי הערך val

שיכלול: distance weighted k-NN

Might want to weight nearer neighbors more heavily . . .

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

נשקלל כל שכן על פי יחס הפוך לריבוע מרחקו מנקודת ה query.

where

$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

כדי לטפל במקרה שבו המרחק 0: לוקחים רק את הנקודות שמרחקן 0 - כי משקלן אין סופי

and $d(x_q, x_i)$ is distance between x_q and x_i

τ קטן: משקל אפסי לנקודות רחוקות.
τ גדול: גם נקודות רחוקות משפיעות

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

שיקלול הדוגמות על פי מרחקן מדוגמת המבחן, מזכיר Locally weighted linear regression (LWR).

Advantages and Disadvantages

Advantages:

- Training is very fast
- Learn complex target functions easily
- Don't lose information

Disadvantages:

- Slow at query time
- Lots of storage
- Easily fooled by irrelevant attributes

בימים בהם היה זיכרון מוגבל: כמעט בלתי אפשרי
היום הרבה יותר סביר: זיכרון זול, דחיסה, מקבול

בעיה: כשיש אלפי features שרובם לא רלוונטיים, יש רעש ואלגוריתמי הקרבה צריכים להתעלם ממה שלא רלוונטי. קללת הממדים! יש פתרונות אבל לא מושלמים: דחיסה, שיכלול פונקציית המרחק.

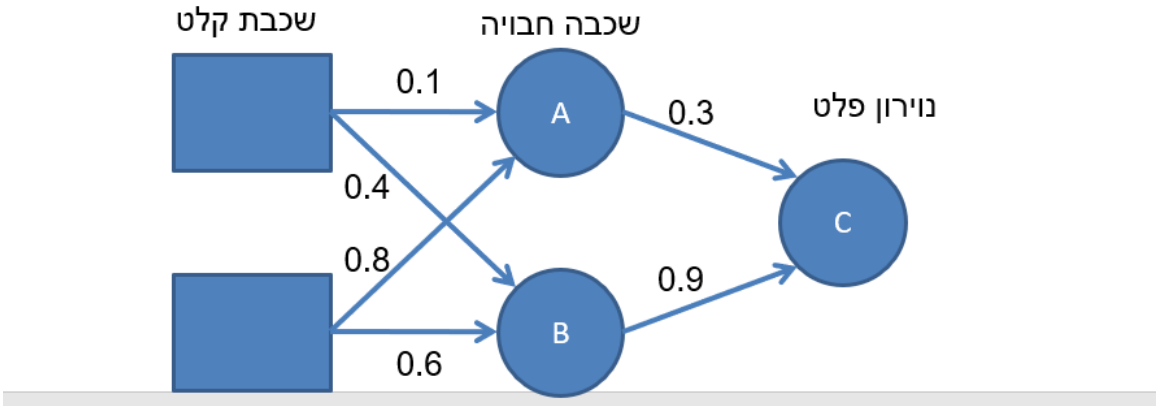
- KNN רגיש מאד לקללת הממדים: המרחק מחושב על סמך כל הממדים (למשל בניגוד לעצי החלטה) למשל: אם יש 20 ממדים ורק 2 רלוונטיים: דוגמאות שזהות ב 2 הממדים הרלוונטיים אבל רחוקות ב- 18, יהיו רחוקות זו מזו.

- פתרונות אפשריים:
  - Feature selection, דחיסת ממדים, טרנספורמציות לממד אפקטיבי קטן
  - בחישוב המרחק: משקל שונה לכל feature: כל ממד נמתח או מתכווץ ע"י מכפלה בקבוע כיווץ (נחליט על הקבועים בעזרת CV ו/או מהיכרות עם D).
  - למרות קללת הממדים, למזלנו בסוגי data מסוימים קימת (ברכת) חוסר האחידות: בהתפלגויות אמתיות שאינם אחידות או נורמליות. למשל, על תמונות וקול, KNN עובד לא רע, למרות הממד הגבוה.

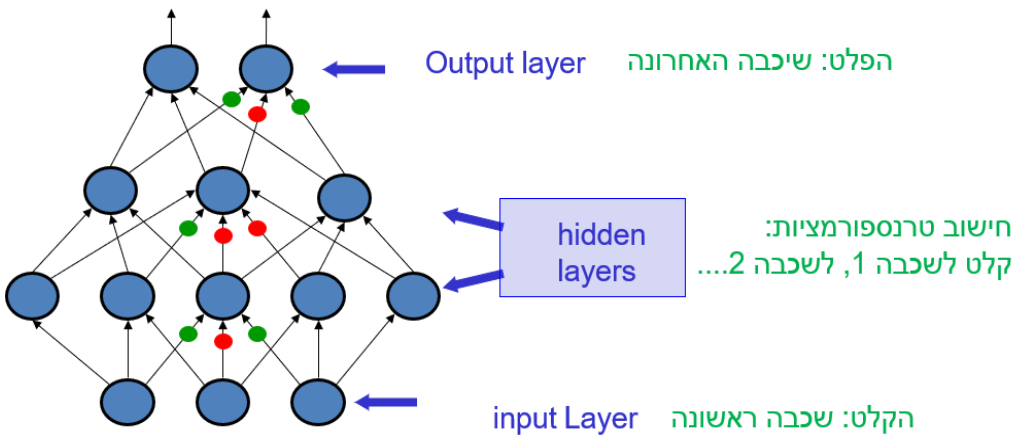
<u>רשתות נוירונים:</u> היפותזות פרמטריות מסובכות באמצעות הרכבה של הרבה פונקציות לא לינאריות ממושקלות. למשל סיגמואידים.

$$y = g(\sum_{j=0}^{m} w_j^{(2)} g(\sum_{i=0}^{n} w_{ji}^{(1)} x_i))$$

GD בעזרת כלל השרשרת (אלגוריתם back-Propagation)



- ארכיטקטורות של רשתות נוירונים: FeedForward Networks:
  - הנוירונים בשכבות הקלט והפלט- **Visible units**.
  - יותר משכבה חבויה אחת: **רשת עמוקה**.
  - ללא שכבות חבויות בכלל: רשת מוגבלת חישובית.
- רשתות רב שכבתיות של יחידות חישוב לא לינאריות מייצרות גבול החלטה לא לינארי.



<u>קלסיפיקציה ביזיאנית:</u>

ההסתברות שבהינתן דוגמה x, היא מקטגוריה k (הא-פוסטריורית)

ה prior של קטגוריה k

ההסתברות של הקלט בהינתן הקטגוריה

$$p(y=k\,|x) = \frac{p(y=k)\,p(x|y=k)}{P(x)}$$

ה prior של x

$$= \frac{p(y=k)\,p(x|y=k)}{\sum_i^K p(x|y=k)p(y=k)}$$

הסתברות הקלט בהינתן הקטגוריה כפול הprior של הקטגוריה

ההיפותזה תמקסם את ההסתברות הזו ונקראת **MAP: Maximal A-Posteriori Estimation**. מכיוון שהמכנה משותף לכל הדוגמות ניתן להתעלם ממנו, הנוסחה:

$$y_{MAP} = argmax_k\,\{p(y=k)\,p(x|y=k)\}$$

Maximum Likelihood: כאשר לא יודעים לשערך את ה- Priors, לפעמים מניחים שהסתברויות ה- prior של הקטגוריות שוות זו לזו ולכן ניתן למקסם רק את p(x|y=k):

$$y_{ML} = argmax_k\,\{p(x|y=k)\}$$

- k הוא קטגורית סיווג (מתוך קבוצה K של קטגוריות).
- X הוא וקוטר של ערכי קלט (מאפיינים) אותו רוצים לסווג.
- y הוא תוצאת החיזוי.

$p(k)\,p(x|k) = p(x,\,k)$= joint probability

כאשר למאפיינים יש הרבה ערכים, או כשהממד גבוה, לא יהיו לנו הרבה דוגמאות ב- D זהות ל- x (אם בכלל), השערוך הישיר להסתברות לא יהיה מדויק, ונזדקק להנחות נאיביות.

- **הנחה נאיבית חזקה: המאפיינים אינם תלויים זה בזה בהינתן קטגוריה i, ואז ניתן להכפיל את ההסתברויות.**

Naïve Bayes Classification (NBC):

$$p(v_1, v_2, \dots . v_n)\ |k) = \prod_{j=1}^{n} p(x_j = v_j | y = k)$$

$$y_{MAP} = \quad argmax_k \{p(k) \prod_{j=1}^{n} p(x_j = v_j | y = k)\}$$

כל מה שנותר זה לשערך את ההסתברויות מתוך D:

- $p(y = k) = \frac{n_k}{n}$
- $p(x_j = v_j | y = k) = \frac{n_{v_{j,k}}}{n_k}$

ה- prior(k): מספר המופעים של קטגורי k מתוך סה"כ המופעים

$P(x_j = v_j | y=k)$:
מספר המופעים מתוך הדגימות של קטגוריה k שבהם (בהינתן ש) מאפיין x_j מקבל ערך v_j

תרגיל: חיזוי בעזרת NBC- האם נשחק טניס?

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

TABLE 3.2
Training examples for the target concept *PlayTennis*.

2 קטגוריות:

- P(yes)≈9/14
- P(no)≈5/14

בהינתן יום חדש x, לא נוכל לשערך את p(x|y=k) ע"י ספירה כי סביר שx לא קיים בD וגם אם קיים, לא יהיו מספק x ים כאלו על מנת לשערך במדויק. נוכל לשערך טוב יותר את ההסתברויות עבור מסוים

שיערוכים להסתברויות המותנות $p(x_i=v_j\ |k)$.

- P(outlook=sunny | yes) ≈2/9
- P(outlook=rain | yes) ≈3/9
- P(Temperature=hot | yes) ≈ 2/9

…….

- P(outlook=rain | no) ≈2/5
- P(Temperature=hot | no) ≈2/5

רוצים לחזות: אם נשחק טניס היום?
x={out=sun,temp=cool,humid=high,wind=strong)
לכל קטגוריה (Yes,No)נחשב את ההערכה ל p(x,k):

עבור yes , נשערך: p(x,yes)

p(yes) p (sun|yes) p(cool|yes) p(high|yes) p(strong|ys)
=9/14 2/9 3/9 3/9 3/9 =9/14 0.0082 = 0.00527

עבור no, נשערך p(x, no)

p(no) p (sun| no)p(cool| no)p(high| no) p(strong| no)
= 5/14 3/5 1/5 4/5 3/5 =5/14 0.0576 = 0.02057

נשווה ונחזיר את הסיווג עם ההסתברות האפוסטריורית הכי גבוהה: **Out = no**

- כדי לקבל ערך ריאלי (ולא הסתברות 0 במידה שחסרים ערכים) נהוג לתקן שיערוכים של ערכי v נדירים בהסתמך על הסתברות ה- prior של v ("כאילו" שקימות עוד מספר קטן m של דוגמאות בקטגוריה שבהן מופיע v). תיקון השגיאות:

כאשר אין "מספיק" דוגמאות המכילות גם **y=k** **וגם** $x_i = v_i$, (למשל כאשר $n_{v_i,k} = 0$)
מוסיפים **m** דוגמאות וירטואליות מקטגורה **k** שמתוכן **m** $p(x_i = v_i)$ הם בעלי ערך $x_i = v_i$.

$$\hat{P}\ (x_i = v_i \mid y=k) = \frac{n_{i,k} + m * p\ (X_i = v_i)}{n_k + m}$$

או אם לא ניתן לשערך את ה priors, (D אינו מייצג אוכלוסייה כללית או הערכים מאוד נדירים,
פשוט מניחים התפלגות אחידה 1/n בין הערכים $v_1 \ldots v_n$.

n_k is the number of examples in class -ל
$n_{i,k}$ = number of examples from class k which has attribute $w_i = v_i$
m is a weight given to examples with prior estimation (number of virtual examples)

Laplace Smoothing

M-estimation לערכים נדירים בהיעדר priors

כאשר נתונים **V** ערכים ועבור חלקם לא ניתן לשערך prior(vi) ע"י ספירה ישירה.
נניח התפלגות אחידה בין הערכים $v_1 \ldots v_n$, ונשערך **prior($x_i = v_i$)= 1/|V|**

אם נוסף **m=|V|** דוגמאות פיקטיביות, המקבלות ערך $x_i = v_i$ בהסתברות 1/|V|
נקבל:

$$P(x_i = v_i \mid y=k) = \frac{n_{i,k} + m(1/|V|)}{n_k + m} = \frac{n_{i,j} + |V|(1/|V|)}{n_k + |V|} = \frac{n_{i,j} + 1}{n_k + |V|}$$

למשל: מילים נדירות שקימות במילון אך לא נמצאות בטקסט ב- D, ישוערכו כ- $\frac{1}{n_k + |V|}$

<u>LDA: Linear Discriminant Analysis</u>: כאשר מאפייני הקלט הם רציפים (שלא כמו ב- NBC) ניתן לפעמים להניח הנחות נאיביות אחרות:

- מניחים התפלגות נורמלית של כל מאפיין (משתנה) בהינתן קטגוריה
- מניחים ווריאנס אחיד לכל התפלגויות המשתנים.

במקרה זה נשתמש בפונקציית הצפיפות (מניחים שנורמלית):

ההסתברות ש- x הוא מקטגוריה k

ה- prior של קטגוריה k

פונקציית הצפיפות של x בהינתן הקטגוריה k

$$p(y = k|x) = \frac{p(y = k) f_k(x)}{\sum_{i=1}^{K} p(y = i) f_i(x)}$$

פונקציית הצפיפות להתפלגות נורמלית חד ממדית:

$$f_k(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x - \mu_i)^2}{2\sigma^2}\right)$$

ניתן לשערך מתוך הדוגמאות ב D:

- variance σ^2 על כל המדגם D.
- תוחלות (ממוצעים) μ_i של הדוגמאות המשויכות לקטגורה k.

חוזים את k בעזרת MAP.

שיערוך הפרמטרים עבור 1D

- מספר דוגמאות האימון המסווגות כקטגוריה k: n_k
- שיערוך התוחלת: ממוצע של כל ה- xים ששויכו לקטגוריה k :

$$\boldsymbol{\mu_k} = \frac{1}{n_k}\sum_{y_i=k} x_i$$

- שיערוך הוריאנס: סטיית התקן בריבוע:

$$\sigma^2 = \frac{1}{n-K}\sum_{k=1}^{K}\sum_{i:yi=k}^{K}(x_i-\boldsymbol{\mu_i})^2$$

- שיערוך ה- priors של הקטגוריות:

$$\mathbf{p(y=k)\approx\frac{n_k}{n}}$$

- LDA מבצע הפרדה לינארית (כמו רגרסיה לוגיסטית) רק שמשערך את המשקולות על פי עקרונות הסתברותיים (נאיביים). שימו לב : במקרה שה- priors של 2 הקטגוריות שווים. גבול ההפרדה הוא באמצע בין התוחלות.

LDA חד ממדי –חיזוי קטגוריה ע"י מיקסום ההסתברויות ושימוש ב Log

צפיפות גאוסיאנית:

$$p(x|y=k) = \frac{1}{\sqrt{2\pi\sigma}}\exp\left(-\frac{(x-\mu_k)^2}{2\sigma^2}\right)$$

חוק ביאס:

$$p(y=k \mid x) = \frac{\frac{1}{\sqrt{2\pi\sigma}}\exp\left(-\frac{(x-\mu_k)^2}{2\sigma^2}\right)P(y=k)}{p(x)}$$

משותף

כאשר מחשבים argmax כדי למצוא את הקטגוריה עם ההסתברות מקסימלית, ניתן להתעלם מביטויים משותפים

$$\text{argmax}_k\{p(y=k|x)\} = \text{argmax}_k\left\{\exp\left(-\frac{(x-\mu_k)^2}{2\sigma^2}\right)P(y=k)\right\}$$

לחילופין, ניתן למקסם את ה log:

$$\text{argmax}_k\{p(y=k|x)\} = \text{argmax}_k\left\{-\frac{(x-\mu_k)^2}{2\sigma^2}+\log(p(y=k))\right\}$$

משותף

אבל:

$$-\frac{(x-\mu_k)^2}{2\sigma^2} = -\frac{x^2}{2\sigma^2}+\frac{2x\mu_k}{2\sigma^2}-\frac{\mu_k{}^2}{2\sigma^2}$$

ולכן:

$$=argmax_k\left\{x\frac{\mu_k}{\sigma^2}-\frac{\mu_k{}^2}{2\sigma^2}+\log(p(y=k))\right\}$$

זה שיערוך לינארית (עבור כל קטגורית סיווג):
בקלסיפיקציה בינארית ניתן להפחית את 2 המשואות הלינאריות ולהשוות ל- 0. מקבלים ישר מפריד:
$z=w_0+w_1x$ נקודות מעל הישר יסווגו כקטגוריה 1 ומתחת לקו , כקטגוריה 2.
דומה מאוד לרגרסיה לוגיסטית, אך ההיפותזה שונה:
ברגרסיה לוגיסטית g(z) וב- LDA על פי נוסחת ביאס.

- בקלסיפיקציה בינארית מאוזנת (כאשר 2 הפריורס שווים), גבול ההחלטה הוא נקודת השוויון x המקיימת את אמצע המרחק בין התוחלות של הפריורס :

$$x = \frac{\mu_2{}^2-\mu_1{}^2}{2(\mu_2-\mu_1)} = \frac{\mu_2+\mu_1}{2}$$

- כאשר יש n משתני קלט, LDA מסתבך קצת :
  - יתכן variance שונה לכל משתנה.
  - תיתכן קורלציה בין המשתנים (נמדדת על פי הקוווריאנס).
  - מניחים שמטריצת ה- variance-co משותפת לכל הקטגוריות (משערכים את המטריצה).

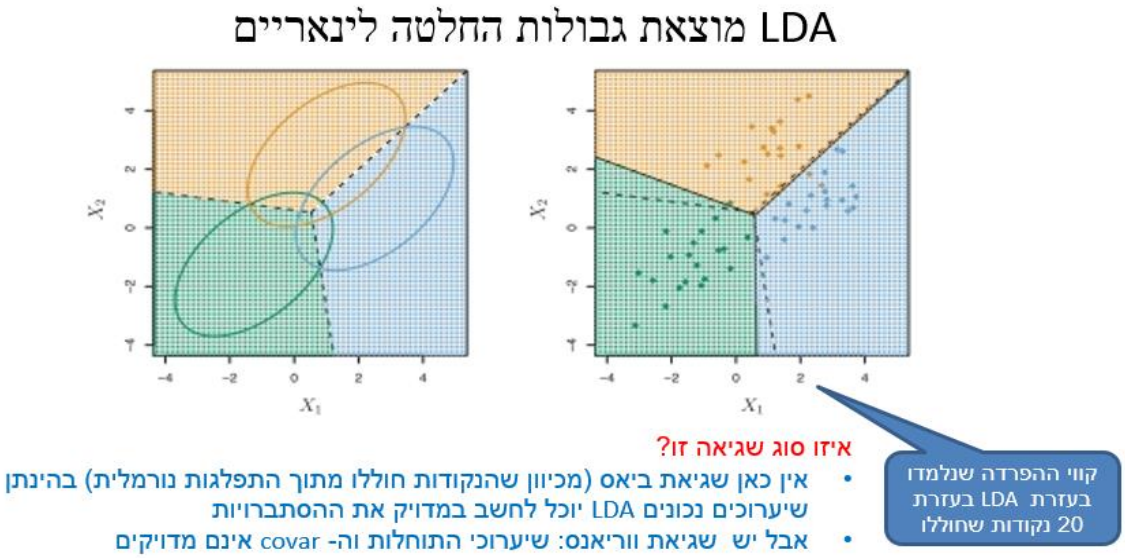
$$f_k(x) = \frac{1}{\sqrt{(2\pi)^n|\Sigma_k|}}\exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\right)$$

Σ_k היא מטריצת ה- covariance המבטאת את הקורלציות בין כל זוגות המשתנים (בהינתן קטגוריה k). ב- LDA, מניחים $\Sigma_i = \Sigma_j$ ז.א. משערכים מטריצת covar משותפת לכל הקטגוריות.

מה שצריך לחשב עבור LDA ב- n משתנים:

- Vector μ_k of n means per category k.
- nxn covar matrix for the entire data.

$$covar(i,j) = \frac{\sum(x_i - \bar{x_i})(x_j - \bar{x_j})}{m}$$

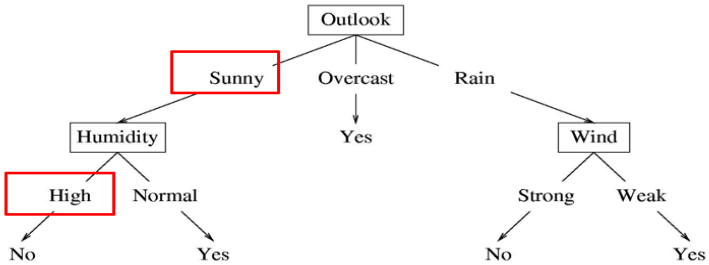


- מתי נשתמש ב- LDA:
  - כאשר X מתפלג נורמלית.
  - הקטגוריות מופרדות היטב.
  - המדגם D קטן ויש חשש ל- overfitting.
  - ניתן לשערך את ה- priors, ברגרסיה לוגיסטית לא ניתן להשתמש בידע המוקדם על priors, מניחים priors אחידים.

עצי החלטה:

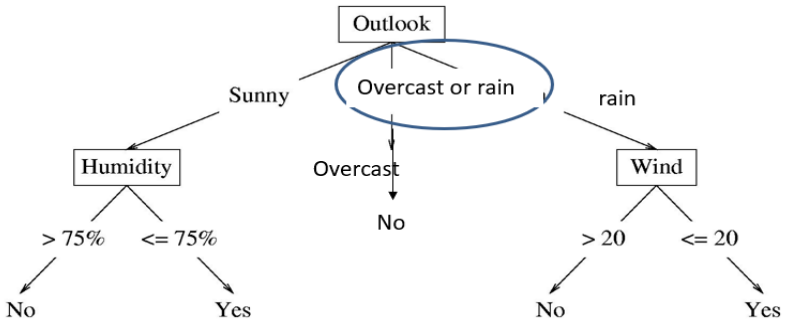
- כל צומת בעץ מבצע פיצול (split) ל- 2 או יותר ילדים.
- צמתי העץ מייצגים feature.
- קשת המחוברת לצומת מייצגת את ההסתברות לערך ספציפי עבור ה- feature אותו מייצגת הצומת.
- העלים מייצגים החלטה סופית (YES\NO).

לדוגמה, חיזוי עבור האם נשחק טניס ביום שמשי לח: NO



ערכים נומרים ניתן לפצל בינארית ע״י בחירת סף (ערך < סף), לחילופין ניתן להגדיר טווחי ערכים (Bins) ולהתפצל כאילו והיו קטגוריות:

בנוסף ניתן לאחד ענפים על מנת להגיע לפיצולים בינאריים:



ניתן לייצג כל פונקצייה בוליאנית כעץ החלטה, כלומר לכל טבלת אמת ניתן לבנות עץ החלטה. מספר הענפים בעץ כמספר השורות בטבלת האמת, אך לעיתים ניתן למצוא עצים חסכוניים יותר.

בהנחה שיש n משתנים בוליאניים, מספר העצים המלאים:
יש $2^{2^{\wedge 2}}$ - עבור כל הפונקציות הבוליאניות.

- זוהי הערכת חסר כי קיימים גם עצים לא מלאים ולכל פונקציה בוליאנית יתכנו מספר עצים.
- כאשר מרחב הקלט איננו בוליאני מקבלים הרבה יותר עצים.
- כאשר מרחב הקלט מכיל שדות נומריים, מרחב העצים הוא אין סופי.

בחירת ה- features ובניית העץ:

אלגוריתם רקורסיבי חמדני ללמידת עצי החלטה (בינאריים)

The same basic learning algorithm has been discovered by many people independently:

$\textsc{GrowTree}(S)$
if $(y = 0 \text{ for all } \langle \mathbf{x}, y\rangle \in S)$ **return** new leaf(0)
else if $(y = 1 \text{ for all } \langle \mathbf{x}, y\rangle \in S)$ **return** new leaf(1)
else
  choose best feature x_j
  $S_0 = \text{all } \langle \mathbf{x}, y\rangle \in S \text{ with } x_j = 0;$
  $S_1 = \text{all } \langle \mathbf{x}, y\rangle \in S \text{ with } x_j = 1;$
  return new node$(x_j, \textsc{GrowTree}(S_0), \textsc{GrowTree}(S_1))$

נתונה קבוצה S של דוגמאות עם מאפיינים בינאריים ומטרה בינארית y=0 או y=1

אם הקבוצה היא הומגנית מבחינת y (ערך אחיד), ניצור ממנה עלה, ונסמנו על פי ה- target. סיימנו

אחרת (S איננה הומגנית): נבחר מאפיין שיפצל "הכי טוב" ל- 2 קבוצות נפרדות. כמה שיותר הומוגניות ב y

נחזיר תת עץ שהשורש שלו הוא השאלה שנבחרה כדי לפצל, ויש לו 2 ילדים שבצורה רקורסיבית ממשיכים להתפצל לעוד קבוצות יותר ויותר הומוגניות עד שהקבוצות "מספיק" הומוגניות או שלא ניתן יותר לפצל

האלגוריתם מסוגל לבנות עץ שיצליח בוודאות להפריד לחלוטין את נתוני האימון.
לחילופין, ניתן להפסיק לפצל אם השיפור בהומוגניות (או בשגיאה) איננו משמעותי.
כך מסתפקים בעצים קטנים יותר. עצים גדולים: פחות ביאס ויותר וריאנס

בוחרים צומת (מאפיין) המפצל את הנתונים לקבוצות בהן סכום השגיאות הכי קטן :

ChooseBestAttribute(S)
choose j to minimize J_j, computed as follows:
$S_0 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 0$;
$S_1 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 1$;
$y_0 =$ the most common value of y in S_0
$y_1 =$ the most common value of y in S_1
$J_0 =$ number of examples $\langle \mathbf{x}, y \rangle \in S_0$ with $y \neq y_0$
$J_1 =$ number of examples $\langle \mathbf{x}, y \rangle \in S_1$ with $y \neq y_1$
$J_j = J_0 + J_1$ (total errors if we split on this feature)
return j

נבחר במאפיין הבוליאני X_j שמפצל את הדוגמאות הרלוונטיות S לצומת הנוכחית בעץ באופן שהשגיאה בקבוצות הפיצול שלו היא הנמוכה ביותר:

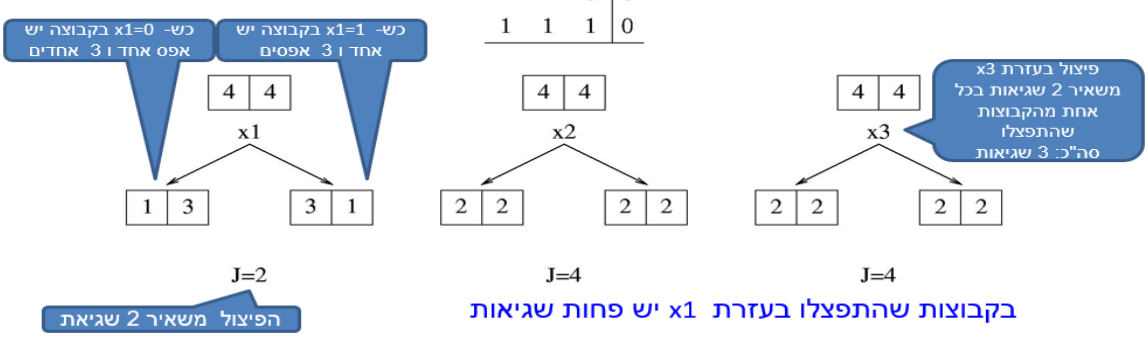
חישוב השגיאה של מאפיין X_j המתפצל ל 2 קבוצות S_0, S_1. סופרים את כמות הדוגמאות בכל קבוצה שאינן תואמות את סיווג הרוב של הקבוצה (y_0, y_1).

השיטה הזו לא עובדת כל כך טוב!

Choosing the Best Attribute—An Example

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

נבדוק לכל המאפיינים את סה"כ השגיאות אחרי פיצול. נבחר במאפיין שמפחית הכי הרבה את מספר השגיאות



- מדד לבחירת פיצ׳רים על פי אנטרופיה רגיש יותר ולכן יכול לכוון יותר את בניית העץ.
- **אנטרופיה:** מודדת את מידת ההומוגניות של קבוצת אירועים.
  - כאשר המשתנה המקרי הוא בוליאני האנטרופיה מקסימלית כאשר הסיכוי ל- 1 הוא חצי. האנטרופיה מינימלית כאשר הסיכוי הוא 1 או 0 (קבוצה הומוגנית).

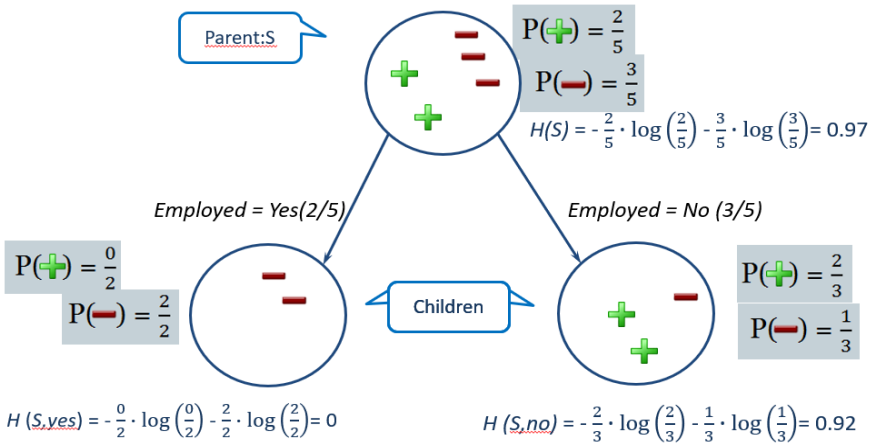
S היא קבוצת כל הדוגמות וההסתברויות נמדדות לפי סכום כל ההסתברויות של כל האפשרויות של ערכים של הפיצ׳ר לו מודדים אנטרופיה.

$$H(S) = -\sum_{i=1}^{N} p_i \cdot \log(p_i)$$

- ניתן גם לחשב אנטרופיה של פיצ׳ר בהינתן ערך מסוים לפיצ׳ר אחר (בעזרת חיתוך בין הפיצ׳רים).

- האנטרופיה H(S) של קבוצה S של ערכי מטרה נמוכה ככל שהקבוצה יותר הומוגנית, לכן נשאף לאנטרופיה נמוכה ככל שניתן.

- **Information gain:** מודד כמה אינפורמציה פיצ'ר מסוים מוסיף, כלומר כמה חוסר וודאות הפיצ'ר מוריד, שיטה **טובה יותר** לבחירת הפיצ'ר הטוב ביותר בכל איטרציה היא בחירת הפיצ'ר בעל ה- IG הגבוה ביותר. כדי לחשב IG של פיצ'ר (צומת בעץ) נדרש ראשית חישוב של האנטרופיה המשוקללת של ילדיו בעץ, כלומר סכום ההסתברויות של כל ילד כפול האנטרופיה שלו (כמה הוא הומוגני).



אנטרופיה משוקללת של 2 הקבוצות שהופרדו:

P(S,Emp=Yes) · H(S,Emp=Yes) + P(S,Emp=No) · H(S,Emp=No)

*=2/5*0+3/5*0.92=0.552*

האנטרופיה המשוקללת של הילדים:

$$\sum_{i=1}^{\#children} P(child_i) \cdot H(child_i)$$

הפיצ'ר שנבחר הוא זה שגורם לירידה הגדולה ביותר באנטרופיה, לכן המדד מחסר את האנטרופיה של פיצ'ר האב שנבדק באנטרופיה של ילדיו, ויבחר הערך הגבוה, אשר יבטיח שנבחר ההורה (פיצ'ר) שהחיסור הנ"ל (IG) מקסימלי.

$$IG(Parent, children) = H(parent) - \sum_{i=1}^{\#children} P(child_i) \cdot H(child_i)$$

תוספת האינפורמציה הנובעת מפיצול קבוצת ערכי מטרה parent לקבוצות ילדים: $S_1 \ldots S_k$.

- IG (parent, children) = entropy (parent) – [P(Emp=Yes) · entropy(Emp=Yes) + P(Emp=No) · entropy(Emp=No)] = 0.97 – (2\5*0 + 3\5*0.92) = 0.42

  Employed feature added 0.42 amount of information, i.e. it reduced the amount of uncertainty (entropy) in target variable, Default (last feature of the example), from 0.97 to 0.55.

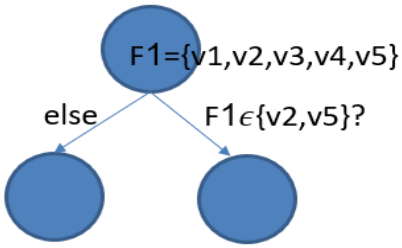
בתהליך בניית העץ: לפני הוספת צומת נבדוק את ה- IG של כל ה- Features שעדיין לא השתמשנו בהם בענף שלו מוסיפים את הצומת החדש. מבין כל ה- Features, נבחר לפצל את ה- Feature עם ה- IG הכי גדול.

פיצול feature שיש לו מספר k ערכים דיסקרטיים

כמה אפשרויות לפיצול:

1. נפצל כל ערך לקבוצה אחרת: נבנה K צמתי ילדים
2. פיצול בינארי ל- 2 קבוצות באמצעות ערך אחד: נחשב IG לכל K הערכים האפשריים
3. פיצול בינארי באמצעות קבוצת ערכים: נאחד כמה ערכים לקבוצה. יש לבדוק IG ל- 2^k תת קבוצות.

אם מפצלים בינארית, אפשר שנשתמש באות מאפיין (בערכים שונים שלו) באותו ענף של העץ



פיצול ערכי REAL: נבדוק את ה- IG למספר ספים שבעזרתם ניתן לבצע פיצול בינארי

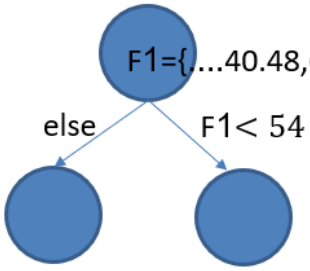
Temp:	40	48		60	72	80		90
PlayTennis:	No	No		Ye	Ye	Ye		No

נסדר את ערכי הfeature בסדר עולה (ברשימה נפרדת את ערכי ה- target המתאימים).
נבדוק את ה -infoGain בנקודות שבהם מתחלף ה- target:

- (48+60)/2=54
- (80+90)/2=84

מובטח ל- Info gain המקסימלי להיות באחד המעברים (פיצול לא במעבר תמיד ייתן פחות הומוגניות מאשר במעבר).

אפשר שנשתמש באותו מאפיין נומרי (בערכים שונים שלו) באותו ענף של העץ



Example: Predicting mortgage default using balance (continuous feature)

Balance:	27K	28K	30K	39K	47K	53K	57K
Default:	Ye	Ye	Ye	No	Ye	No	No

מהו ה- IG עבור Balance>50 ?

H (parent) =

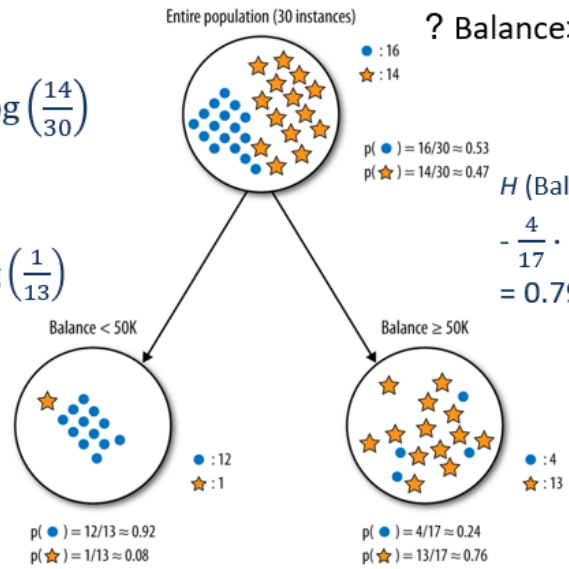
$$-\frac{16}{30}\cdot\log\left(\frac{16}{30}\right)-\frac{14}{30}\cdot\log\left(\frac{14}{30}\right)$$

= 0.99 (very impure)

H (Balance < 50K) =

$$-\frac{12}{13}\cdot\log\left(\frac{12}{13}\right)-\frac{1}{13}\cdot\log\left(\frac{1}{13}\right)$$

= 0.39



H (Balance ≥ 50K) =

$$-\frac{4}{17}\cdot\log\left(\frac{4}{17}\right)-\frac{13}{17}\cdot\log\left(\frac{13}{17}\right)$$

= 0.79

$IG = H\text{ (parent)} - [P(C_L)\cdot H(C_L) + P(C_R)\cdot H(C_R)\,] =$

$$0.99 - \left[\frac{13}{30}\cdot 0.39 + \frac{17}{30}\cdot 0.79\right] = 0.37$$

מדד אי שוויון : Gini Index Measure
(דומה לאנטרופיה)

$$G = \sum_{i=1}^{N} p_i\cdot(1-p_i)$$

$$H = -\sum_{i=1}^{N} p_i\cdot\log(p_i)$$

עבור משתנה אקראי בינארי:

שוויון בין קבוצות כאשר הן הומוגניות:
H=G=0

אי שוויון בין קבוצות כאשר הטרוגניות:
G= ½* ½ + ½* ½ =1/2
H=1/2*1+1/2*1=1

- כאשר הסתברויות לקטגוריות קרובות ל- 1 או ל- 0, שני המדדים יתנו מספר נמוך (מתקרב ל 0 מלמעלה).
- הפונקציות מקבלות ערך גבוה כאשר יש הרבה קטגוריות בהסתברות זהה :
  - G מתקרב ל- 1 : N(1/N)(1-1/N)=(N-1)/N
  - H יכול להגיע למספרים גבוהים מאוד : - N(1/N)log(1/N)=-log(1/N)

- עצי החלטה יכולים להפריד כל קבוצת דוגמאות באמצעות גבול החלטה מלבני (מדרגות).

בעצי החלטה ניתן רק לחתוך אופקית או אנכית. ניתן לקרב כל פונקציה (כך שתתבצע הפרדה מושלמת) בעזרת קירוב של חיתוכים אופקיים ואנכיים. אבל קירוב כזה בהרבה מיקרים אינו מכליל היטב- שגיאת ווריאנס. למשל, נצטרך הרבה data כדי שהחיתוכים יתקרבו לקו ישר.

עבור רגרסיה

- ה- MSE של הדוגמאות בקבוצה נלקח כמדד לשיפור (במקום אנטרופיה).
  ה- GAIN של כל split מחושב ע״פ השיפור ב- MSE בעקבות הפיצול.
- הצומת תיבחר על פי מיצוע של כל ה- MSEים של הילדים.
- ממוצע ה- targets של הדוגמאות הנופלות בקבוצה (צומת) נלקח כחיזוי ה- y של הקבוצה, לכן כל עלה חוזה את הערך הנומרי הממוצע של הדוגמאות שבעלה.

העץ שנוצר **לא** תמיד אידיאלי, מכיוון שלעיתים נבחרים פיצ׳רים רדומים אשר לא משפיעים על הפיצול\רעש. במקרה כזה לעיתים לא ניתן להבדיל האם הפיצ׳ר הוא parity (אינו תורם\רעש) או שכן.

- עצים זה מודל עם שגיאת ביאס נמוכה ושגיאת ווריאנס גבוהה.

טיפול בבעיית הווריאנס:

- עצירה מוקדמת - מפסיקים להצמיח עלים לפני שהגענו לעלים הומוגניים:
  - כאשר השיפור באנטרופיה אינו משמעותי.
  - כאשר השיפור בשגיאת הווליד‌ציה איננו משמעותי.
- רגולריזציה Complexity pruning: הענשת השיפור באנטרופיה לעצים גדולים או צמתים עמוקים InfoGain/TreeSize.
- לחילופין: מגדלים את העץ במלואו ואז מקצצים (Tree Post Pruning): למשל מחפשים תת-עץ עם פחות שגיאת ווליד‌ציה. כבד יותר חישובית אבל ביצועים טובים יותר.
- Ensemble: Bagging, Boosting, Random Forest.

Pros

- Very illustrative, very easy to explain to non-experts.
  Some believe that trees more closely mirror human decision making
  Easy to understand rules can be deduced from the tree
- Flexible enough to approximate any function
- Ranks importance of features
- Can be implemented with easy to understand rules
- Can be scaled to big data
- Long history including many extensions: CART (77), ID3 (83), C4.5 (94), C5(2000?), Sprint, VFDT

Cons

- High-variance- non-robust: easy to overfit
  small changes to data may lead to large change in tree
  Use Pruning (with regularization) and Ensembles to overcome
- Can be unnecessary Big
- Inductive Bias (towards shorter trees)
- Greedy search: InfoGain or Ginie – not always detect the best possible tree

- **Bagging: Bootstrap Aggregation** (עבור עצי החלטה)

נבצע bootstrapping (דגימה עם חזרות, מופיע לעיל) כדי לגדל B עצים שונים
מ- B מדגמים ששונים קצת זה מזה.

נמצע את תוצאות החיזוי של העצים השונים:

- ברגרסיה ממצעים את החיזוי.
- בקלסיפיקציה משתמשים ב- majority vote.

נהוג לגדל עצים לא מקוצצים: לכל עץ שגיאת וואריאנס גבוהה אשר מצטמצמת ע"י ה- ensemble.
בפרקטיקה מגדלים אפילו אלפי עצים ומחברים לפרוצדורת חיזוי אחת.

העצים דומים מידי אחד לשני ולכן שגיאת הווריאנס **אינה** יורדת משמעותית מכיוון שדגימות ה-bootstrapping הינן קורלטיביות והצמתים (המשמעותיים) בסמוך לשורש, נשארים לרוב קבועים במקומם בעץ.

כאשר לוקחים מדגם (אקראי עם חזרות) של N דוגמאות מתוך N, יהיו דוגמאות out-of-bag – שאינן מופיעות במדגם.

ניתן לראות שכאשר דוגמים N, בערך $((N-1)/N)^N=1/e=0.368$ מהדוגמאות הם out-of-bag.

לכל דוגמא בקבוצת האימון ממצעים את N החיזוי של כל העצים שלא התאמנו על אותה דוגמה.
השגיאה הממוצעת של תחזיות ה- out-of-bag היא הערכה טובה לשגיאת הטסט.

אפשר לחשב Out of Bag error Estimation בצורה יעילה יותר מ- CV והשיטה נותנת הערכות שגיאה דומות.

- **Random Forests: bagging & bootstrap with random feature sets**

עצים שגדלים על מדגמים דומים עלולים להיות דומים מבחינת המאפיינים שבהם משתמשים.

נרצה לגדל עצים הנבנים מקבוצות שונות של מאפיינים.

- נבנה עץ לכל מדגם בשיטת bootstrap.
- בכל פעם שבוחרים צומת לפיצול, נגביל את המאפיינים לבדיקה ל- m'<m מאפיינים הנבחרים רנדומית מתוך m המאפיינים שלרשותנו.

ללא מגבלה על בחירת המאפיינים, סביר שברוב העצים היה נבחר אותו מאפיין (דומיננטי ב- IG) עבור השורש. ביער רנדומי: המאפיין ה"חזק" ביותר לא יילקח בחשבון בסיכוי של (m-m')/m.

- ב- bagging ללא מגבלת מאפיינים, אכן רואים כי העצים קורלטיביים.
- ב- random Forest העצים פחות קורלטיביים, ככל ש- m' קטן יחסית ל- m בד"כ נקטין את הווריאנס ונגדיל את הביאס.

הבחירה במגבלת m' מאפיינים היא היפר-פרמטר:

- m=m' מייצר bootstrap bagging רגיל.
- מקובל m'=m^(1/2).

כאשר יש הרבה מאפיינים והם קורלטיביים אחד עם השני, יבחר m' << m (הרבה יותר קטן מ).

והאלגוריתם עבור בחירת הפיצ'ר הטוב ביותר ב- RF:

```
ChooseBestFeature(S,Fset, m') returns (BestFeature, UpdatedFeatureSet)

If Fset is empty return (Null,Null)

Randomly select m' features from Fset (if |Fset|<m', select all features in Fset)
Find the feature BestF with the best IG
        remove from Fset,
        return(BestF, Fset)
```

<u>Support Vector Machine (SVM):</u> מאד מזכיר רגרסיה לוגיסטית. מונחים קשורים:

- **Maximum Margin Classifier:** קלסיפייר בינארי לינארי על בסיס מקסום המרווח (margin) בין הקטגוריות. דורש כי הנתונים יהיו ניתנים להפרדה לינארית. כלומר, בהינתן קבוצת אימון D של דוגמאות מסווגות בינארית {+1,-1} מממד n,
הניתנת להפרדה לינארית, רוצים למצוא היפר-מישור מממד n-1 המפריד בין הדוגמאות. בממד n ההיפר-מישור המפריד הוא אילוץ מהצורה: $h(x)=w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = 0$
כל נקודה שמקיימת את האילוץ, היא נקודה על המישור. נקודה x מעל המישור תסווג 1, ומתחת 1- בהתאם לסימן של x לאחר הצבתה במישור (אם מתקבל ערך קטן מ- 0 אז מתחת למישור = 1-, אחרת, מעל = 1).

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: $y_i\, h(x_i) = yi\ (w_0 + w_1x_{i1} + w_2x_{i2} + \cdots + w_nx_{in}) > 0$
המרחק של נקודה x מהמישור נותן אינדיקציה לרמת הביטחון בסיווג:

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

ישנם אין סוף מישורים המקיימים את האילוצים (לכל נקודה). ברגרסיה לוגיסטית בחרנו מישור מפריד שממזער את ה CE, עבור SVM נבחר את המישור בעל ה- margin המקסימלי.

ה- Margin מוגדר כרוחב שבו ניתן להרחיב את ההיפר-מישור המפריד לפני שנתקל בנקודת אימון. נקודות האימון הראשונות שנתקלים בהן נקראות support, והן קובעות את רוחב ה- margin על ידי כך, שהמישור חוצה בדיוק באמצע בין support שלילי לחיובי.

המישור המפריד עם ה- margin המקסימלי הוא הטוב ביותר מפני ש:

- יכולת הכללה טובה יותר עבור נקודות טסט המתקרבות למישור המפריד.
- אינו רגיש למיקום רוב הנקודות שמחוץ ל- margin ולכן פחות Overfitting.

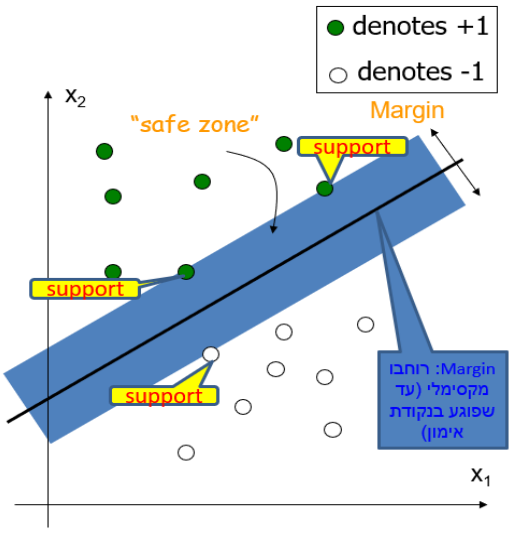
כאשר נדרשים למקסם את ה- margin נותרים אך ורק עם מישור אחד כזה- מקסימלי.

הדבר נעשה ע"י בעיית אופטימיזציה תחת אילוצים:

- $Maximize\ M$
  $M, w_0, w_1, w_2, \dots, w_n$
- subject to $y_i\ (w_0 + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_n x_{in}) \geq M$
  $for\ every\ \ (x_i, y_i) \in D$
- Subject to $\sum_{j=1}^{n} {w_j}^2 = 1$

יש אין סוף מישורים שמקימים את התנאי (מכפלות בקבוע).

- בוחר מבין כל המישורים כזה עם נורמל יחידה.
- נותן משמעות ל$w_0 + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_n x_{in}$: המרחק של הנקודה מהמישור.
- בדומה לרגולריזציה, אילוץ על גודל המשקולות.



- כאשר הנקודות אינן ניתנות להפרדה לינארית- אין פתרון לבעיית האופטימיזציה.
- תוספת של נקודה אחת עלולה לגרום לתזוזה דרסטית של המישור המפריד
  ול- margin צר מאוד, אינדיקציה ל- Overfitting. הגדרת הבעיה אינה מאפשרת outliers בכלל. גם אם ניתן להפריד, **ישנה רגישות גבוה לנקודות שבסמוך למישור המפריד**.
- ה- MMC נקרא קלסיפייר קשה\כבד.

- **Support Vector Classifier = Soft Margin Classifier:** הרחבה לקלסיפייר בינארי לינארי שאינו דורש הפרדה לינארית מוחלטת של דוגמות האימון. 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 ל– OutLiers בודדים.
  - מספר רב יותר של נקודות support– פחות רגישות לתזוזות ונקודות חדשות.

נחפש **Soft** Margin Classifier שאינו מכריח כל דוגמא להיות מעבר ל- margin אלא מיעוט של דוגמאות יכולות להופיע בצד הלא נכון של ה- margin מבלי להשפיע עליו (להצר\להרחיב אותו) או על מישור ההפרדה. הקלסיפייר ה"רך" יסווג את רוב הדוגמאות ברמת ביטחון גבוה (מחוץ ל margin רחב), חלק מן הדוגמאות יסווגן נכון אבל בתוך ה- margin וחלק מהן יסווגו באופן שגוי.

הדבר נעשה ע"י:

נוסיף Slack variable ε_i לכל דוגמה, אשר מאפשר לה להופיע בצד הלא נכון.

אם $\varepsilon_i = 0$ הדוגמא, תהיה מחוץ ל- margin ותסווג נכון.

אם $1 \geq \varepsilon_i > 0$ הדוגמא תהיה בצד הלא נכון של ה- margin אבל בצד הנכון של המישור (עדיין תסווג נכון).

אם $\varepsilon_i > 1$ הדוגמא תהיה בצד הלא נכון של המישור המפריד ותסווג לא נכון.

Maximize M

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

Subject to: $for\ every\ (x_i, yi) \in D$

המכפלה מקטינה בפועל את ה- margin ואפילו הופכת אותו לשלילי אם הדוגמה בצד הלא נכון של המישור

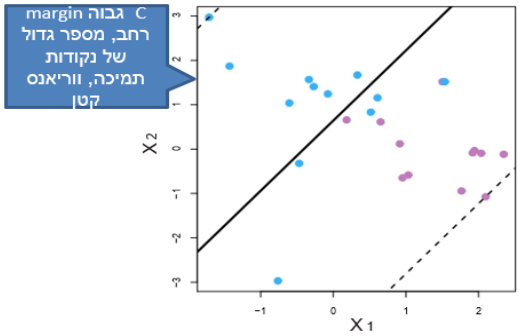
- $y_i\,(w_0 + w_1x_{i1} + w_2x_{i2} + \cdots + w_nx_{in}) \geq M(\mathbf{1} - \boldsymbol{\varepsilon_i})$
- $\boldsymbol{\varepsilon_i} \geq \mathbf{0},$
- $\sum_{i=1}^{m} \boldsymbol{\varepsilon_i} < \mathbf{C}$
- $\sum_{j=1}^{n} {w_j}^2 = 1$

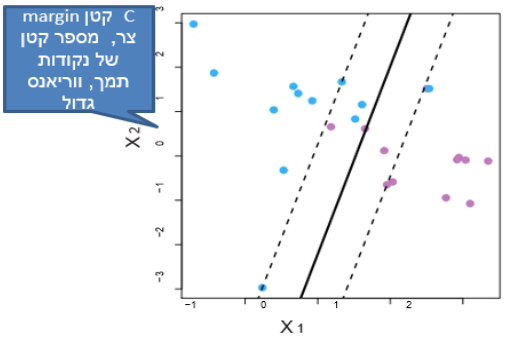
C היפר פרמטר המגביל את סכום החריגות מ- M

C שולט ב Bias-variance Tradeoff

C : תקציב עבור חריגות מה Margin:

- לא יתאפשרו יותר מ- C חריגות מהמישור המפריד כך ש: $\varepsilon_i > 1$.
- C=0: אין כלל תקציב: כל הדוגמאות צריכות להימצא מהצד הנכון של M, $\varepsilon_i = 0$.
- נקודה בצד הנכון של המישור אבל בתוך ה- margin תיקח מהתקציב: $0 < \varepsilon_i < 1$ בהתאם למרחקה מהמישור.
- נקודה בצד הלא נכון של המישור תיקח מהתקציב נתח גדול יותר: $\varepsilon_i > 1$ ותסווג לא נכון.
- C גדול, מאפשר להרבה נקודות לחרוג מה- margin וכל נקודה שחורגת היא נקודת תמך.
  - כשיש הרבה וקטורי תמך יש פחות רגישות לתזוזות בנקודות אלו, או להוספת נקודות תמך חדשות (שגיאת וריאנס קטנה יותר). מצד שני, C גדול מאפשר סיווג לא נכון של דוגמאות אימון ולכן מעלה שגיאת ביאס.





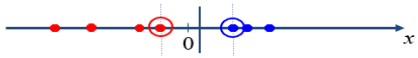
- העובדה שרוב הנקודות נמצאות מחוץ ל- margin ולכן אינן משפיעות על קביעת המישור נותנת יתרון בהורדת וריאנס מול שיטות הרגישות לכל הנקודות יחד כמו LDA ועצים.

- ברגרסיה לוגיסטית ניתן להראות שמצב דומה מתקיים, אינטואיציה: גם שם נקודות רחוקות כמעט ולא משפיעות על גבול ההחלטה. יש דמיון רב גם בהתנהגות האמפירית של 2 השיטות.

- **SVM = Soft Max Margin:** הרחבה לקלסיפייר בינארי לא לינארי ע"י טריק הקרנל.

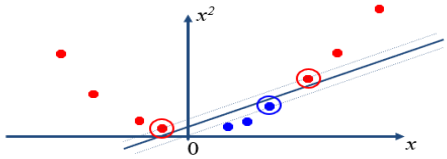
Non-linear SVMs

- Datasets that are linearly separable with noise work out great:



- But what are we going to do if the dataset is just too hard?

- How about… mapping data to a higher-dimensional space:



- General idea: the original input space can be mapped to some higher-dimensional feature space where the training set is separable.

הדבר יעשה ע"י:

- אפשר שנוסיף polynomial features ונבצע את האופטימיזציה במרחב מממד גדול יותר.
- אפשר להשתמש במאפיינים לאו דווקא פולינומיאלים, 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 יעיל מבחינה חישובית.

ניתן להוכיח (לא בקורס) כי ניתן להביע את ההיפותזה הלינארית באמצעות מכפלות לינאריות של הווקטור הנבדק עם כל אחת מנקודות האימון ב- D (מכפלה פנימית).

נחפש בעיית אופטימיזציה דואלית: במקום לחפש וקטור W שימקסם את M ויקיים את שאר האילוצים של SVC, נחפש וקטור של אלפות שיעשה אותו דבר.

- נראה כי חישוב בעזרת האלפות לא יעיל (כאשר m>>n) אך יש לזכור כי רק האלפות של נקודות התמך יילקחו בחשבון. האלפות של הנקודות שאינן תומכות, יתאפסו.

את המכפלה הפנימית נוכל להחליף בהכללה שלה (קרנל): $K(x_i, x_{i'})$

- **קרנל** היא פונקציה שמכמתת דמיון בין 2 וקטורים. הפונקציה צריכה לקיים תכונות של קרנל- לצרכים פרקטיים כמעט כל פונקציית דמיון תהיה בעלת תכונות אלו. קיימים הרבה כאלו, דוגמאות:
- הקרנל הלינארי (Pearson Measure), מודד קורלציה בין וקטורים מנורמלים והוא מאפשר רק הפרדה לינארית:

$$\text{Linear Kernel}=K(x_i, x_j)=\sum_{k=1}^{n} x_{i,k}x_{j,k}$$

- הקרנל הפולינומיאלי, מעביר את הבעיה למרחב בממד גבוה יותר הכולל מאפיינים פולינומיאלים מדרגה d. כך SV classifier פותר את הבעיה במרחב הגדול יותר (n^d) מבלי לבצע בפועל את הטרנספורמציות. אך התוצאה אקוולנטית להתמרת כל הנקודות למרחב הגדול ואז הפרדה לינארית ע"י SVC רך:

ההיפותזה: $h(x)= \alpha_0 + \sum_{i=1}^{m} \alpha_i K(x, x_i)$ **(d>1)**

$K(x_i, x_{i'})=(x_i \cdot x_{i'})^d= (\sum_{j=1}^{n} x_{ij}x_{i'j})^d$

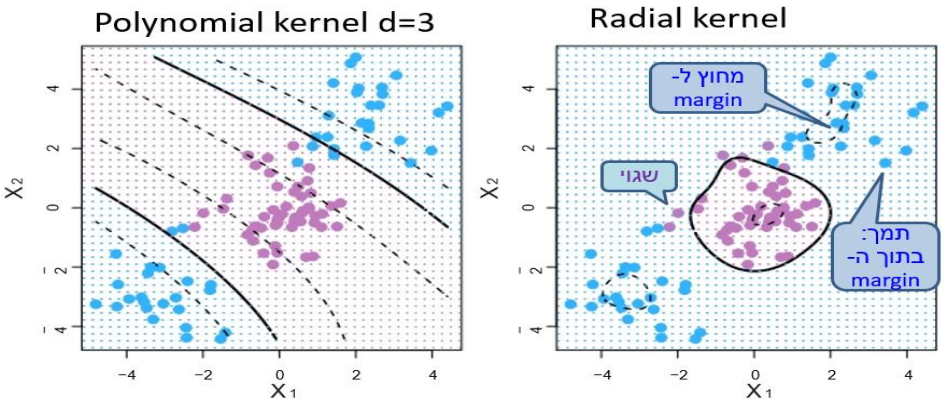
קיימת וואריאציה $(1+x_i \cdot x_{i'})^d$

- Radial (Gaussian) Kernels : דומה לדמיון אאוקלידי (בצורת פעמון) רק שהמרחק מושפע אקספוננציאלית (הפוך) : ככל שהנקודות רחוקות זו מזו, הדמיון נהיה מזערי. תכונת לוקאליות (המתוגברת באמצעות פרמטר גמא גדול) גורמת לכך שנקודות רחוקות לא ישפיעו כמעט כלל.
  - ככל שהסיגמה קטנה, הגאוסיאן צר יותר (low-bias, High-variance).
  - ככל שהסיגמה גדולה, גאוסיאן רחב– דומה למרחק אאוקלידי (High-bias).

ההיפותזה: $h(x)= \alpha_0 + \sum_{i=1}^{m} \alpha_i K(x, x_i)$

Radial Kernel of $\gamma>0$:

$K(x_i, x_{i'})= exp(-\gamma \sum_{j=1}^{n}(x_{ij}-x_{i'j})^2)$



יתרונות שימור בקרנלס :

- אין צורך לעבור למרחב הגדול. מספיק לחשב את הדמיון בין כל זוגות הנקודות. המעבר למרחב הגדול נעשה באופן לא מפורש implicit. יתרון גדול חישובית כי בהרבה אפליקציית המרחב הגדול הוא עצום. ה- Kernel מאפשר ל- features לעבור טרנספורמציה, אבל נשארת הפשטות והקלות של העבודה הלינארית אחרי הטרנספורמציה.
- בזמן אימון צריך לחשב את הדמיון של כל הזוגות אך בזמן חיזוי לנקודה x מספיק לחשב את הדמיון שלה רק לווקטורי התמך.
- לפעמים המרחב המורחב הוא מממד אין-סופי ואז אין בכלל אפשרות להעביר את הבעיה למרחב הגדול בצורה מפורשת- כמו במקרה של radial kernels.

- **דרך אחרת להסתכל על מקסום ה- margin : קיבוע ה- margin ומזעור משקולות.**

ניזכר במטרה : רוצים למקסם את ה- margin. תמיד אפשרי להגדיל את המשקולות ולקבל margin גדול יותר, אבל מה שחשוב זה ה- margin יחסית ל- norm. לכן שומרים על $\|w\|=1$. **לחילופין** : במקום למקסם את ה- margin, נוכל לשמור אותו קבוע בגודל M=1, אך נמזער את המשקולות $\|w\|$.

נתבונן באלגוריתם לקיבוע ה- margin ומזעור משקולות :

- Instead: Fix margin, minimize weights
- **Minimize** $\|w\|$
  Subject to $y_i(w \cdot x_i) \geq 1, \quad \text{for all } i$

- **Hinge Loss with L2 Regularization :** פונקציית loss שממזערת חריגות מ M=1.
  - כש- y*h(x) גדולה מ- 1 ה- loss 0.
  - בתוך ה- margin, הloss בין 0-1.
  - אם הנקודה בצד השני של גבול ההחלטה, loss>1.

בעזרת לגרנגיאן (כלי לפתירת בעיה עם אילוצים) ניתן להפוך את האילוצים לפונקציית loss ולהראות כי בעיית האופטימיזציה הקוודרטית שקולה למזעור 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

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

Similar to Logistic regression with Ridge regularization.

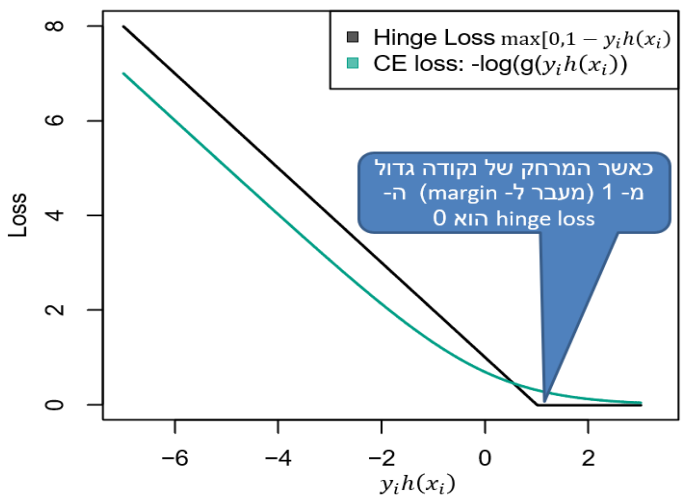
SVM with Gaussian kernel is a soft approximation to KNN.

Support Vector Machine: workflow:

1. Choose a kernel function.
2. Choose a value for C.
3. Solve the quadratic programming problem (many software packages available).
4. Construct the discriminant function from the support vectors.

קשר הדוק בין SVM לרגרסיה לוגיסטית

Hinge Loss ≈ CE loss



שימור ב- SVM יעיל כאשר:

- בעיה מממד גבוה.
- מעט נתונים: סכנת high-variance.
- רוצים לבדוק מהר גבולות החלטה לא ליניאריים הודות לזמינות גבוהה של ה- Kernel Trick.

אבל: קיימות בעיות ביצועים כאשר יש הרבה וקטורי תמך.

SVM עבור יותר מ k>2 קטגוריות: SVM נבנה עבור קלסיפיקציה בינארית ולא ניתן להרחבה בקלות ליותר מ2 קטגוריות.

- One VS One: בנה k(k-1) קלסיפיירים: אחד לכל זוג קטגוריות. בהינתן דוגמא x הפעל את כל הקלסיפיירים. פלט החיזוי: הקטגוריה בעלת רוב החיזויים.
- 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.
- **השפעת עליית מספר הממדים על מספר וקטורי התמך:** מספר וקטורי התמך יעלה מכיוון שצריך יותר נקודות כדי לתמוך בצינור ה- margin הרב ממד. בממדים קטנים נראה למשל בסביבות 1% וקטורי תמך מסה"כ נקודות האימון. בסביבות 100 ממדים נתחיל לראות 40%, 60% ואפילו 80%. זוהי השפעה על ה- scalability של SVM.

Summary: Support Vector Machine

1. Support Vector Classifier (Soft Max Margin)
   - Better generalization ability & Controlling bias-variance tradeoff
   - Can be extended for regression

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 when many support vectors
   - Does NOT extend easily to multi-class
   - Similar to other well established algorithms:
     - Logistic regression (possible with kernels)
     - Weighted-KNN)

Unsupervised Learning:

שיטות אישכול- Clustering:

1. גילוי ידע: הבנת קשרים ודמיון בין נקודות הנתונים:
   - מציאת דמיון בין נקודות.
   - ויזואליזציה.
   - סגמנטציה.
   - מדידת מרחק בין קבוצות של נקודות: למשל כמה רחוקה קבוצת הפרימטים מהלטאות (מתי חל הפיצול?).
2. איתור חריגות/שגיאות.
3. זיהוי דמיון תצפית לקבוצות ב- D. למשל זיהוי דובר בקבוצת דוברים (שזהותם איננה ידועה).

בהינתן קבוצה D של דוגמאות אימון (ללא תיוגים), רוצים לחלק לקבוצות זרות שמכסות את D ומכילות דוגמאות "דומות".

Clustering לא מוגדר במדויק כמו Supervised learning. לא תמיד ברור מה רוצים להשיג:

- לא ברור כיצד מודדים "דמיון", שיטות תלויות אפליקציה.
- לא ברור כיצד מערכים את התוצאות. ישנן המוני שיטות לקלסטרינג- אין שיטה אחת שמתאימה לכל.

K-Means Clustering: נתונה קבוצת דוגמאות D מממד n ונתון מספר ה- clusters (אשכולות) K. נחלק את D ל- K קבוצות זרות המכסות את D. נרצה כי הדמיון בין הנקודות בתוך כל cluster יהיה הגדול ביותר, ז.א. השוני (אי-הדמיון) בין הנקודות שבכל אשכול יהיה מינימלי.

אחת השיטות הפופולריות למדידת השוני בתוך קלסטר **Within-Cluster-Variation**, מתבססת על סכום ריבועי המרחקים בין הנקודות שבקבוצה: נסכם לכל נקודה את סכום המרחקים בינה לבין כל שאר הנקודות בקלסטר ונמצע (למשל, מרחק אאוקלידי). נרצה למצוא קלסטרים שהסכום הכולל של השוני- WCV של כל אחד מהם הוא מינימלי. פונקציית המטרה בהמשך לכן Total WCV:

WCV: Within-Cluster-Variation

$$WCV(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} d(x_{i,}, x_{i'})^2$$

$$d(x_{i,}, x_{i'}) \overset{2}{=} \sum_{j=1}^{m} (x_{i,j} - x_{i',j})^2$$

$$minimize_{c_1 \ldots c_k,} \left\{ \sum_{k=1}^{K} WCV(C_k) \right\}$$

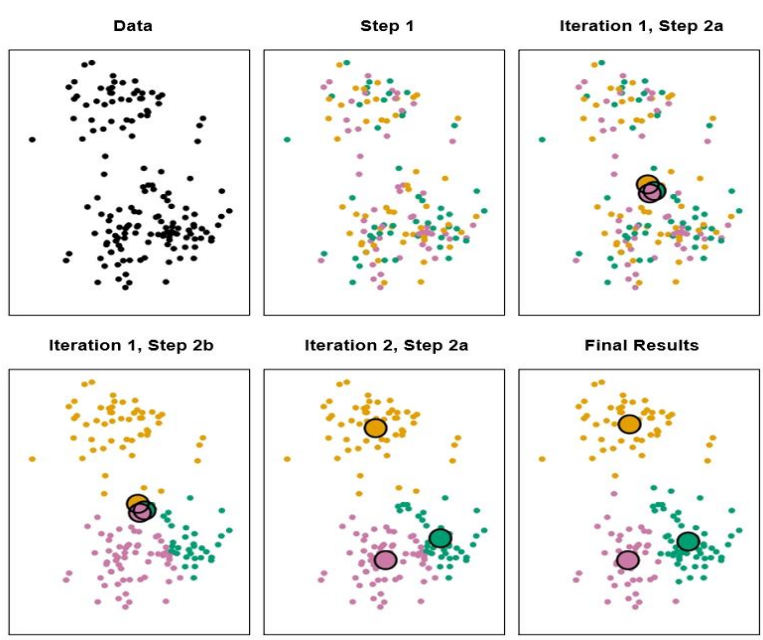
$$TWCV = \sum_{k=1}^{K} WCV(C_k)$$

Algorithm K-Means Clustering

1. Initial cluster assignment:
   Randomly assign a cluster number (1…K) for each example x ∈ D
2. Repeat until cluster assignment stop changing
   a) For each cluster: compute its centroid:
      The vector of m feature means
   b) Re-assigne each example x in D to the cluster whose centroid is the closest

1. מצרפים רנדומית כל דוגמא לאחד מ- K הקלסטרים
2. חוזרים שוב ושוב עד שאין שינוי בקלסטרים:
   a) מחשבים את המרכז לכל קלסטר.
   b) מצרפים מחדש את הנקודות על פי קירבתן למרכזי הקלסטרים.

- מרכז הקלסטר הוא הווקטור של ממוצעי המאפיינים של הדוגמאות שבקלסטר.
- אם נשתמש בנורמה d לחישוב המרחק בין 2 וקטורים, ניתן להוכיח כי בכל איטרציה, המרחקים בתוך כל קלסטר $WCV(C_k)$ אינם גדלים ולכן בכל צעד ניתן שמבצעים בו שינוי, משפרים .



***חשוב לציין, הסנטרואיד הינה רק שיטה אחת מתוך כמה למדידת שוני בין קלאסטרים, עוד בהמשך.**

K-means מבצע ירידה בפונקציית המטרה WCV עד לקבלת מינימום מקומי. ניתן לראות כי קיים קשר בין ה-WCV לבין סכום המרחקים ממרכז הקלסטר:

$$\frac{1}{|C_k|}\sum_{i,i\prime\in Ck}\sum_{j=1}^{m}(x_{ij}-x_{i'j})^2=2\sum_{i\in C_k}\sum_{j=1}^{m}\ (\mathrm{x_{ij}}-\bar{x}_{kj})^2$$

בכל איטרציה המרחק בין הנקודות יורד מונוטונית: מחשבים סנטרואידים חדשים ואז מכניסים לקלסטר את הנקודות הקרובות לכל כל סנטרואיד. אם התווספה נקודה לקלסטר אחד הרי שהנקודה יצאה מקלסטר שני ואז מרחקה ממרכז הקלסטר החדש קטן ממרחקה ממרכז הקלסטר הישן. לפיכך, ה- WCV של הקלסטר הראשון גדל פחות מההקטנה ב- WCV שקרתה בקלסטר השני ומתבצעת ירידה בפונקציית המטרה TWCV עד למציאת מינימום **מקומי**- מסקנה: רצוי לנסות את האלגוריתם כמה פעמים (התחלה רנדומית שונה) ולבחור WCV מינימלי, מפני שניסיונות הרצה שונים עלולים לתת נקודות מינימום שונות.

- חיסרון גדול של K-means: יש לקבוע מראש את מספר הקלסטרים.
- **קלאסטרים היררכיים- DendoGrams:** * נקודות דומות מתמזגות מוקדם, נקודות שאינן דומות מתמזגות מאוחר יותר.
  - **מלמטה למעלה:** מתחילים כשכל דוגמא ב- D היא קלסטר (עלה בעץ). בכל איטרציה מאחדים 2 קלסטרים שהם הכי קרובים עד שמקבלים בקלסטר אחד את D.
  - **מלמעלה למטה:** מתחילים מקלסטר בודד D ומפצלים ל- 2. כך שוב ושוב עד שמקבלים קלסטרים בני דוגמא אחת.

פירוש דנדוגרם **Bottom-Up**: עץ- עליה מהעלים לכיוון השורש מראה באיזה סדר מתמזגים הקלסטרים. בגובה 0 כל קלסטר מכיל עלה יחיד, כל מיזוג מוסיף לגובה. ככל ש- 2 קלסטרים מתמזגים מוקדם יותר- הם דומים יותר אחד לשני. הגובה מייצג את אי-הדמיון בין הדוגמאות הממוזגות בקלסטרים, ע"י חיתוך מלמעלה בגובה המתאים, נוכל לקבל כל מספר של קלסטרים שנרצה.

אלגוריתם קליסטור Bottom-Up: נתונה פונקציית "אי דמיון" d(x, y) בין 2 דוגמאות ובעזרתה נגדיר "אי דמיון" בין קלסטרים Inter-Cluster-DisSimilarity=ICD(C_1, C_2) המודד כמה 2 קלסטרים שונים זה מזה (יש כמה ווריאציות).

1. הכנס כל דוגמא $x \in D$ לקלסטר נפרד
   חשב את מדד אי-הדמיון לכל $\binom{n}{2}$ זוגות הדוגמאות.
2. עבור i=n,n-1,...2: (עד שכל הדוגמאות מאוחדות בקבוצה אחת)
   a) בחר זוג קלסטרים בעלי אי-דמיון מינימלי ומזג אותם לקלסטר אחד
   b) חשב את אי הדמיון של הקלסטר הממוזג מול כל אחד מהקלסטרים שנותרו
   c) גובה הקלסטר הממוזג בדנדוגרם הוא השוני WCV(C) בין הדוגמאות ששויכו לקלסטר

שיטות למדידת "אי-דמיון" בין קלסטרים

אי הדמיון $ICD(C_1,C_2)$ בין קלסטרים מודד כמה 2 קלסטרים שונים זה מזה
Inter-Cluster-DiSimilarity

ישנן וואריאציות שונות שכולם מתבסוסות על מדידת Dissimilarity בין 2 דוגמאות:

Linkage (between clusters) types:

1. Maximal (complete) Dissimilarity:
   - Compute all pairwise dissimilarities d(x,y) between x∈C1 and y∈C2
   - Output the Maximal dissimilarity

Maximal and Average are most popular and typical generate balanced dendograms

2. Average
   - Compute all pairwise dissimilarities d(x,y) between x∈C1 and y∈C2
   - Output the average of the dissimilarities

3. Minimal (single)

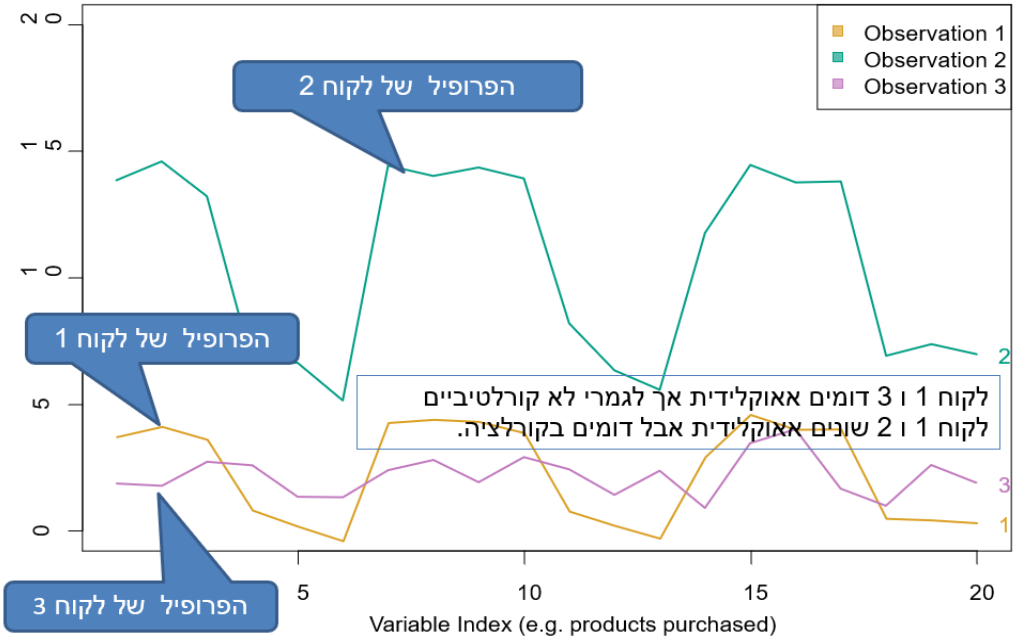
Not so popular: tend to generate imbalanced dendograms

   - Compute all pairwise dissimilarities d(x,y) between x∈C1 and y∈C2
   - Output the Minimal dissimilarity

Used in Genomics- problems with visualization: The height of a merged cluster is below the height of its constituents

4. Centroid
   - Output the dissimilarity d(center(c1),center(c2)) between the centers of the two clusters. Usually the center of a cluster is the average point.

Thera are many choices of Dissimilarity measures, by the necessity: for example Euclidian Distance against Correlation-based distance



החלטות שצריך לקבל כשמבצעים קליסטור:

1. באיזה "מרחק" אי-דמיון נשתמש.
2. האם ננרמל את המאפיינים וכיצד.
3. כמות הקלסטרים שנרצה לאתר (למשל עבור k-means).

עבור קליסטור היררכי:

1. באיזה Inter cluster linkage.
2. היכן לחתוך את הדנדוגרם.

קילסטור לא עובד טוב כאשר:

אין הפרדה טובה בין תת הקבוצות.

- נפח תת הקבוצות שונה משמעותית (כאשר תת קבוצה אחת דלילה משמעותית עלולים לא למצוא אותה כלל ולעוות את הקלסטרים הפחות דלילים).
- קיימים מספר קטן (ולא ידוע) של תת קבוצות אמתיות, אך יש מספר קטן של outliers שלא שייכים לשום תת-קבוצה אמיתית. הקליסטור עלול להקצות קלסטרים גם ל- outliers ובכך לעוות את תת הקבוצות האמתיות. שיטת קליסטור שנקראת Mixture of models מבצעת קליסטור "רך":
  - פחות רגישה ל- Outliers.
  - פועל היטב גם כשיש חפיפה גאוסיאנית בין תת הקבוצות.