**סימונים:**

, – Data, - פרמטרים, פקטור נרמול  
, , קונסיסטנטיות: ∀ i=1,…m,

**הסתברות**  
התפלגות משותפת היא ההתפלגות המשותפת לקבלת כל סיווג של (חיובי, שלילי וכו...) והיא נתונה ע"י: כאשר מסווג או שלילי או חיובי ו- היא ההסתברות לאותו סיווג.  
כעת נוכל לסווג תוצאה חיובית בהינתן ע"י חוק בייס: .  
,

התפלגות גאוסיאנית , כש- התוחלת ו- מטריצת ה-Covariance; אז הצפיפות היא:

**Proper Kernel**

*הגדרה:*

1. סימטרי:
2. חיובי: לכל ולכל , לכל כש-.

סגור לחיבור, כפל, כפל בסקלר חיובי, הרכבה עם פולינום

ניתן להכליל את הגאוסיאני עם מטריקה אחרת.

**מודל PAC:**הנחה כי ההתפלגות קבועה אך אינה ידועה, הדוגמאות נשלפות i.i.d.**,** בנייה של חוק החלטה ולא הסתברות.בנייה:

1. ישנה התפלגות משותפת לא ידועה .
2. פונק' המטרה כאשר: .
3. השערות כאשר .
4. השגיאה .
5. ישנו "אורקל" אשר מייצר לנו דוגמאות .

הגדרה:

1. ו- עם מחלקות קונספטים מעל .
2. היא PAC learnable על ע"י אם:
   1. קיים אלגוריתם A כך ש:
      1. עבור התפלגות מעל ו- ועבור כל .
      2. המוצא הוא השערה עם הסתברות ושגיאה .

מחלקת השערות סופית (Realizable case) :

נאמר כי הוא משערך רע אם .

אלגוריתם:

* נדגום מדגם S עם m דוגמאות.
* נמצא כך שהיא קונסיסטנטית – מסווגת את כל הדוגמאות נכון.

האלגוריתם נכשל אם היא .

נתקן השערה שהיא . ההסתברות ש- קונסיסטנטית:  
 .

מחלקת השערות (Non Realizable case):

מה קורה כאשר , יש להגדיר את המטרה מחדש: תהיה בעלת השגיאה הכי קטנה:  
 , המטרה: .

ERM ALGORITHM:

1. עבור כל נסמן את כשגיאה על מדגם S.
2. נחשב את ההסתברות ש: עבור כל הדגימות, סכימה על m דגימות ושימוש בחסם צ'רנוב נקבל כי החסם: עבור כל הדגימות בהשערה אחת
3. כעת עבור כל ההשערות ב- נקבל: וגודל המדגם הדרוש:

מסקנה: .

סיכום PAC:

עבור Realizable case: .

עבור Non Realizable case: .

**VC Dimension:**הגדרה: C מחלקת קונספטים ו- S הוא המדגם. ההקרנה של C על S: .ניתוץ: נאמר ש C מנתצת את S אם .VC-dim מוגדרת כקבוצה הכי גדולה אשרש עבור S מתנתצת: . אם לא קיים מקסימום כזה נאמר שה- VC-dim הוא אינסוף.

**Rademacher Complexity:**המטרה היא למצוא חסם הדוק יותר ללא תלות בהתפלגות. נסמן:  
 ו: .הגדרה: עבור מגדם S בגודל m:   
ו- .  
כעת, ה- overfitting הצפוי:   
הוכחה: נוסיף מדגם S’ עבור ובכך הוא יהפוך ל ובכך יש לנו פעמיים שזה קטן מפעמיים .  
כעת, בהסתברות ועבור כל :  
 .

**Perceptron:**מודל Online: מתקבלת דוגמא, האלגוריתם מעריך את הסיווג הנכון, האלגוריתם מקבל את הסיווג האמיתי ומתקן בהתאם. אפשר להניח ש- ע"י הוספת מאפיין שהוא קבוע ומשקלו . באלגוריתם הזה כל עוד אין שגיאה המפריד נשאר זהה. בנוסף ננרמל את כל הנקודות .

1. . (או .
2. בהנתן דוגמא להעריך חיובי אמ"מ .
3. בשגיאה: , כש- הסיווג המוערך.

**משפט:** עבור סדרת דוגמאות קונסיסטנטית עם המפריד כש- וקטור יחידה. אז מספר השגיאות חסום ע"י , כש- (ה-margin).

**Unrealizable case:**

Assume:

Hinge loss =

טעות לפי ה-margin מתקבלת כאשר . המרג'ין נכון כאשר ה-loss שווה ל-0.

אלגוריתם:

1. נאתחל את המשקולות .
2. בהינתן דגימה נסווג אותה כחיובית אם"ם .
3. כאשר יש שגיאה בזמן t נעדכן את המשקולות: כלומר:
   1. שגיאה לגבי דגימה שלילית (): .
   2. שגיאה לגבי דגימה חיובית (): .

Margin Perceptron:  
*אם נניח שאפשר להפריד את ה-Data עם Margin אז מספר השגיאות של האלגוריתם הבא חסום ע"י :*

אותו דבר כמו Perceptron, אבל נבדוק אם וגם אז נבצע (כמו בשגיאה).

**SVM:**לכתוב KKT.  
ה SVM בונה [על מישור](https://he.wikipedia.org/wiki/%D7%A2%D7%9C-%D7%9E%D7%99%D7%A9%D7%95%D7%A8) (Hyperplane) שהוא המפריד הלינארי (מפריד את המרחב לשני [חצאי מרחבים](https://he.wikipedia.org/wiki/%D7%97%D7%A6%D7%99_%D7%9E%D7%A8%D7%97%D7%91) שכל אחד מהם אמור להכיל בעיקר דוגמאות מסוג  אחד), וכן שני על מישורים מקבילים לו, אחד מכל צד, במרחק זהה, אשר מתלכדים עם דוגמת אימון אחת מכל מחלקה. מטרת ה SVM היא להגדיל את המרחק בין המפריד הלינארי ובין כל אחד מהמישורים המקבילים (מרחק המכונה שוליים), ולמעשה למצוא את המפריד בעל השוליים הרחבים ביותר. דוגמאות האימון המתלכדות עם מישורי השוליים נקראות וקטורים תומכים, ומכאן נגזר שם האלגוריתם.המסווג SVM הוא הפתרון ל: **.**נשתמש בלגרנג'יאן על מנת למצוא את הפתרון:

כשגוזרים לפי אז מורידים את הסכום ויוצא שכל יהיה שווה ל α.  
אם הוא וקטור תומך אז: .  
איך מוצאים את ו- ? פותרים את הבעיה הדואלית: נציב את מה שמצאנו ע"פ הנגזרות החלקיות ונקבל כי:   
ולכן הבעיה הדואלית היא: .  
ע"מ למצוא את b, נוכל לחשב (KKT אז ).  
בסוף הבעיה הדואלית להוסיף: ואז להפוך את ואז: .

כעת נוסיף אילוץ אשר "משלם" עבור הפרות של המפריד הלינארי והבעיה החדשה:  
כאשר אפשר לכתוב את כ: ולפתור את הבעיה הדואלית בהצבה הזאת. (HINGE LOSS).  
Positive Semi-Define Matrices (PSD):  
קרנלים קשורים באופן ישיר ל PSD. מטריצה היא PSD אם:

* היא סימטרית.
* .
* כל הערכים העצמיים הם אי-שליליים.
* קיימת מטריצה .

מטריצה סימטרית אם:

* הערכים העצמיים ממשיים.
* הערכים העצמיים הם בסיס אורתוגונאלי: .

עבור מטריצה עם וקטורי עמודה , מטריצה אלכסונית עם ערכים עצמיים באלכסון, נקבל: .

כל וקטור ניתן לרשום כ: : .

מסקנה: המטריצה *היא* PSD.

SVM קרנל:  
העברת הדגימות למימד גבוה יותר בעזרת פונקציה יכולה להיות בעיה קשה, אבל בצורה הדואלית מתייחסים לנקודות רק בעת חישוב מכפלה פנימית. החישוב ייקח אז רק . Kernels:

Polynomial:

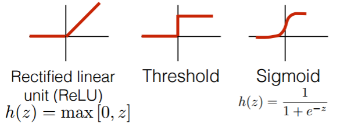
Gaussian:

Intersection:

ב- SVM משתמשים בעיקר בקרנל גאוסי: . ואז קו ההפרדה: אם מציבים w ופותרים מקבלים סכימה עם הגאוס שזה קו הגובה של צירוף לינארי של כל הנקודות המסווגות כפי שרצוי (מרכז הגאוס).

**SGD for SVM*:***

1. *דוגמים דגימה באופן אחיד מהמדגם.*
2. *אם אז: .*
3. *אחרת: כאשר: .*

**Neural Network:***המודל מכיל שכבות ובכל שכבה צמתים: .  
נשתמש ברשתות נויירונים ע"מ לבנות מסווג בינארי: או מסווג רב-מימדי: .*Activation function:   
*כל צומת מקודדת על מישור במרחב כאשר השכבה הראשונה היא על מישור מעל הווקטור קלט .*VC dimension *עבור רשתות נויירונים: נניח כי ישנם E משקולות שכל אחת מקודדת ע"י D ביטים. החסם על VC הינו: וכאשר המשקולות ממשיות: .  
Training Neural Nets: נשתמש ב SGD ונרצה להביא למינימום:* ***.  
Back propagation:***

1. *Forward pass: עבור כל שכבה נחשב את המוצא הלינארי ופונק' ההפעלה הלא לינארית .*
2. *נחשב את ווקטור "השגיאה" עבור כל שכבה מהאחרונה לראשונה:*

***עצי החלטה:***

*בעץ החלטה, כל צומת פנימית היא היפוטזה או והעלים הם הסיווגים השונים.  
המטרה: עץ החלטה קטן המסווג את כמעט כל הדוגמאות נכון.*

*VC dimension לעצי החלטה: כאשר לעץ S עלים ומספר n מאפיינים.  
כאשר המאפיינים בינאריים:   
חסם תחתון: עבור S דוגמאות, נבנה עץ החלטה כאשר כל דוגמא מגיעה לעלה מסוים ולכן .  
חסם עליון: מספר העצים הינו מספר קאטאלן: עבור כל עץ: לכל צומת יש מאפיינים והלייבל עבור עלה הוא . מספר הפונקציות: ולכן נקבל:  
 .*במקרה הכללי כאשר המאפיינים לא-בינארים: .  
עיקרי האלגוריתם:

1. תהליך רקורסיבי:
   1. נבחר מחלק למידע בשורש.
   2. נחלק את המידע באמצעות השורש.
   3. נבנה תת עץ ימני עבור .
   4. נבנה תת עץ שמאלי עבור .

זמן הריצה:   
במקרה הגרוע: במקרה הממוצע: .  
פונק' פוטנציאל:   
אלגוריתם לבניית העץ:

1. אם כל הדוגמאות ב S בעלות אותו סיווג b, ניצור עלה עם הסיווג b ונחזיר את העץ.
2. עבור כל נחשב כאשר:  
    .
3. יהיה .
4. נחלק את S באמצעות ל- .
5. כעת נחזור רקורסיבית ל- 1עם תתי הקבוצות .

קטימה של העץ (Pruning):  
נקבל כקלט עץ T ומדגם S ונוציא עץ T’ כאשר בקטימה הבסיסית נחליף צומת פנימי בעלה ובמקרה המתקדם נחליף אותו באחד מיילדיו.  
Reduced Error Pruning: נחלק את המדגם כאשר עם החלק הראשון נבנה את העץ ובחלק השני נשתמש ע"מ לקטום עת העץ. כעת עבור כל צומת פנימי: נבדוק את השגיאה על אותו צומת ועל מסוים ואם השגיאה של העלה קטנה מזה של הצומת, נחליף את הצומת בעלה.

עבור כל העץ, מחשבים את סכום הפוטנציאל על העלים:

הפוטנציאל תמיד יקטן בכל שלב. סיווג מושלם:

**מאפיינים של הפוטנציאל**:

*,* .

*הפוטנציאל הוא הגבול עליון של הטעות.*

האלגוריתם הנאיבי יוביל ליצירת עץ גדול עם 0 שגיאות על דוגמאות ה-train, דבר המעלה את הסיכוי ל-overfitting. לכן יש לחתוך את העץ לאחר היצירה.

**אפשרויות חיתוך**:

1. *מחלקים את ה-train ל-2. עם חלק אחד בונים את העץ ועם השני חותכים אותו. בודקים עבור כל צומת פנימי האם יש פחות טעויות במצב הנוכחי או כאשר אחד הבנים שלו (עלה) יחליף אותו.*
2. *Model selection: חסרון השיטה הוא זמן ריצה גבוה יחסית O(m2). יוצרים m חיתוכים שונים ובכך גם m עצים שונים. מוצאים את החיתוכים שמביאים את הטעויות למינימום (ניתן להשתמש בשני מדגמים כמו ב-1). ניתן לבחון גם באמצעות SRM (kd הוא גודל החיתוך עם d טעויות):*

**Weak Learning**:  
בהינתן אלגוריתם עם ביטחון ושגיאה גרועים: , אפשר לשפר את השגיאה.  
**AdaBoost**: נבנה מסווג לינארי באמצעותת weak learner, בכל פעם נוסיף עוד מסווג ונתאים לדגימה S. בכל צעד t, יש היפוטזה , נבחר התפלגות של המדגם. נמצא את המסווג ביחס להתפלגות. נוסיף את המסווג למחלקת ההיפוטזות, נחשב את ו- ונחשב את .

חשוב לזכור:

**אלגוריתם:**

1. יש מחלקה של weak learners H: .
2. אתחול: בהינתן מדגם נקבע את ההתפלגות לכל דגימה: .
3. עבור כל צעד :
   1. נקבל בהתאם ל- .
   2. נגדיר: , .
   3. נגדיר: .
4. תחזית: .

אינטואיציה:

1. איך אנחנו משנים את ההתפלגות:
   1. שגיאה המשקולות גדלות.
   2. לא שגיאה המשקולות קטנות.
   3. התמקדות בדוגמאות "קשות".
2. מהם הפרמטרים:
   1. מחלקת ה- weak learners H.
   2. מספר האיטרציות T.

השגיאה של האלגוריתם בהינתן השגיאות של כל איטרציה:

**Regression and PCA:**

המטרה היא למפות קלט X ללייבל Y כאשר Y הוא בינארי או multiclass באמצעות פונק'. ההפסד הצפוי: כאשר באה ממחלקת ההשערות F.  
נניח כי הדוגמאות באות מהתפלגות ונרצה להביא למינימום: .

כיוון שאנחנו לא יודעים את הערכים הצפויים, נמצא קירוב:  
 .  
בעיית ה- ERM: **.**

נתחיל בפונק' לינארית: נניח כי , כאשר .   
תמיד: .  
נוסיף bias: .

**Logistic Regression:**\* מחזירה מסווג לינארי. דגימה לא ניתנת להפרדה ע"י מסווג לינארי. אם יש נקודה המסווגת "+" בצירוף של הקמור של שלוש נקודות המסווגות "-". אם יהיה אפשר להפריד את המדגם, כל מה שבקמור מסווג באותו סימן.  
\* נתונה בעיית רגרסיה משהו: פונק' האופטימיזציה היא פונק' ה- LOSS של הנקודות: ואז המשקולות: כדי למצוא את המשקולות גוזרים את הפונק' ומשווים לאפס.  
\* אם לא נתונים X במפורש אלא אז נוכל לומר כי כשנרצה למזער לפי a שהוא המשקולות של X.

**Solving Linear Regression:**  
נפתור: **.**נגדיר: , .  
נרצה להביא למינימום: .  
גרדיאנט: **.**נשתמש ב: .  
נציב שהגרדיאנט שווה אפס ונקבל: . אם הפיכה נקבל: .  
נגדיר ונקבל: המודדת את הקורלציה בין i ל- j. עבור משתנים עם תוחלת אפס, נקבל כי זו מטריצת הקווואריאנס.

מה קורה אם לא הפיכה? יש תלות בין הפיצ'רים ואז נקבל אינסוף פתרונות ל .  
פתרון: Regularized Regression: נוסיף פקטור אשר יהיה נמוך עבור פתרונות שנרצה וגבוה עבור פתרונות שלא נרצה. אפשרויות:  
Ridge: נרצה להביא למינימום: . הנגזרת כמעת זהה למקרה הרגיל:  
(ההפיכה הזו תמיד מתקיימת). כעת נקבל כי:   
אם : אחרת: .

**Linear Subspaces:** תת-מרחב במימד r מוגדר ע"י: . התת-מרחב הוא כל הנקודות כך שקיימים: ו- כך ש הם ייצוג r מימדי של .  
אם: ו- נקבל כי: .  
נניח בלי הגבלת הכלליות שהבסיס אורתוגונלי: (נוכל תמיד לקבל ע"י Gram-Schmidt)  
אז: , .

הקרנה לתת-המרחב: בהינתן נקודה שלא בתת-מרחב נמצא את הנקודה שבתת-המרחב:  
**PCA:**\*\*\* כאשר עושים טרנספורמציה אז מטריצת D ו- U יכולות לשנות את המערכת. D יכולה לשנות כל כאורדינטה בנפרד. U יכולה לגרום לשינוי של מערכת הצירים. צירופים של סקלר או מכפלה בסקלר לא יגרמו לשינוי.

נניח בה"כ שתוחלת הנקודות היא 0. אם נטיל את המידע למימד נמוך, ונחזיר חזרה, נאבד חלק ממנו. נרצה למזער את האבדן. ראינו שזה שקול למיקסום השונות. כלומר הוקטורים בעמודות. קידוד: ; פענוח: . זו הטלה למרחב שנוצר ע"י וקטורי .

כש- ע"ע וו"ע מסודרים בגודל יורד. נסמן ונקבל LP:

והפתרון

השונות עליה אחראי כל PC מחושבת ע"י:

ניתן לבצע PCA על ידי חישוב הע"ע והו"ע של XXT. יותר יעיל לחשב בעזרת SVD.

חישוב ה-PC ה-i:

**נקודות חשובות**:

1. חשוב לבצע נרמול של הנתונים לפני חישוב ה-PCA. חוסר נרמול מוביל לאי אחידות בהשפעה של כל פי'צר.
2. נקודות חריגות יכולות להשפיע מאוד על נכונות ה-PCA. מאחר והחריגות רחוקים משאר המידע, הם משפיעים לרעה יותר מאשר הנקודות החיוביות. לפעמים ה-PC הראשונים יכולים להתקבל עקב חריגים. ניתן לנסות להפתר מחריגים אלו על ידי regression של ה-PCs הראשונים.

**Probabilistic PCA**

לא ידועה rank מלא, כש-.

לכל W נקבל פתרון לפי רגרסיה: . לפי SVD ו- היא d העמודות הראשונות של . אז בדיוק כמו PCA.

**Kernel PCA**

נניח פירוק SVD, ו- מטריצות מצומצמות ל-d הערכים הסינגולריים הגדולים ביותר. . אם ננסה לעשות PCA ישירות נצטרך לחשב את שיכולה להיות גדולה מאוד. אז כש-,

**בעיית PCA:** המטרה היא למצוא תת-מרחב הקרוב ביותר למידע:  
 זוהי בעיית האופטימיזציה ל PCA.  
לפני הרצת PCA יש להפוך את התוחלת של המידע לאפס כיוון שזה האופן האופטימאלי למצוא תת-מרחב "מחובר" וגם כי כך מטריצת הקורלציה הופכת למטריצת הקווואריאנס.  
נקבע: והקלט החדש: .

**פתרון PCA:** מטריצת הקווואריאנס: . הערכים העצמיים והווקטורים העצמיים:   
אז הפתרון ל- PCA הוא: , בעיקרון: ניקח את r הווקטורים העצמיים עם הערכים העצמיים הגדולים ביותר.

**קלסטרים: K-Means:**  
נניח כי מספר הקלסטרים ופרוטוטייפס הוא K והפרוטוטייפס: . הנקודות בקלסטר ה- i צריכות להיות קרובות ככל הניתן ל- .נגדיר את המרחק בין נקודה לפרוטוטייפ: .  
והמטרה היא למצוא המביא למינימום את: . האלגוריתם קטן בכל איטרציה.

אלגוריתם: נעדכן בכל איטרציה את: , נשייך את כל הנקודות לקלסטר הקרוב אליהן ביותר ונעריך  
מחדש את להיות התוחלת של כל הנקודות ששויכו לקלסטר i. נחזור על הפעולה עד להתכנסות.

המטרה: נכתוב את המרחק כ: כאשר הוא המשתנה indicator ה-"בוחר" את הקלסטר הטוב ביותר לנקודה i.  
 בכל איטרציה נעדכן את:  
   
נקבל: בכל איטרציה הפונק' קטנה.

**Special Clustering:**  
באופן כללי נתון גרף עם משקלים אי-שליליים על הקשתות, שמוגדרים לפי מטריצת השכנויות הם הדמיון בין שני קודקודים. הדרגה של קודקוד . נגדיר . הלפלסיאן: ו-

מחפשים את כך ש- וממזער את . הע"ע המינימלי לא מעניין כי הו"ע המתאים לו הוא   . אז לוקחים את הו"ע עם הע"ע השני הכי קטן.

**Learning Generative Module:**

KL Divergence: בהינתן שתי התפלגויות רוצים לדעת כמה הן קרובות.  
ישנם מספר אפשרויות למדוד את המרחק ביניהם:   
אנו נשתמש ב- Kullback-Leibler (KL) divergence: .

תכונות של KL Divergence:

1. לא שלילית (הוכחה: השתמש ב-   
   ו: ).
2. לא סימטרית.
3. לא חסומה (נקבע את ו- עבור כלשהו).

עבור כל נבדוק כמה הם קרובים, תחילה לפי p ואז לפי q.

גישת ERM: כך נוכל למקסם את ההערכה האימפירית.

Maximum Likelihood: בהינתן מידע, כדי למצוא את ההתפלגות נפתור: זה נקרא maximum likelihood estimation.

גאוסיין עם תוחלת ושונות לא ידועים: בהינתן סקלר x ו- F הם כל   
הגאוסיאנים: (לטטא במכנס להוסיף את המימד אם מדובר בווקטור)  
 בעיית ה- ML: .  
ונקבל: עבור התוחלת האמפירית ו- עבור השונות האמפירית.

**Gaussian Mixture Modules (GMM):**

1. Z נבחר רנדומלי לפי .
2. בהינתן התוצאה נדגמת לפי גאוסיין .
3. ההתפלגות של X היא: .

Maximum Likelihood for GMM:

1. ה- GMM: .
2. נניח כי קיים מידע .
3. נרצה למצוא את הפרמטרים עם ML. ונפתור: .
   1. לא כדאי להשתמש בנגזרת כיוון שאין פתרון טוב.
   2. נשתמש ב- Expectation Maximization EM.

**אלגוריתם EM:** האלגוריתם משפר את ה-"דמיון" בכל איטרציה ומתכנס למקסימום מקומי.  
לכל נקודה קיים המתאר לאיזו קבוצה הוא שייך. אנחנו לא יודעים את אך אנחנו יכולים לחשב את השערוך על הפרמטרים כדי לחשב: . ובכך נחשב את שהוא ההסתברות שאכן הנקודה בקלסטר הזה: וכיוון ש- זוהי הסתברות נקבל: .

העדכון של התוחלת: כאשר k הוא מספר הקלסטרים.  
העדכון של השונות: כאשר k הוא מספר הקלסטרים.

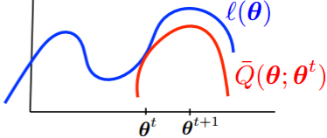
אנחנו מעוניינים לפתור את הבעיה אך אנחנו לא יודעים את z אך אנחנו יודעים את שנותן מושג לגבי z   
כלומר: .  
ה- "דעה הקדומה" שלנו נותנת לנו את המשקולות על z.  
נחשב את הממוצע של ה- Complete likelihood עם   
המשקולות: .  
נניח כי ולכן זה נראה כי אנו רואים: .  
את אפשר לראות כקירוב של ה- likelihood. ולכן המקסם אותו.

**האלגוריתם:** ל EM יש שני צעדים: לשערך את ה- likelihood (E step) ולמקסם אותו (M step).

על כל איטרציה t:

1. E-step: .
2. M-step : .

EM משפר את ה- likelihood:

1. נטען כי ונגדיר: .
2. צעד זהה ל- M-step: .
3. Lemma: אז נקבל:
   1. .
   2. .   
      

**Maximum Likelihood (ML) Estimators**

פרמטרים שמגדירים היפותזה, -Data

Likelihood Function:

Log-likelihood:

Log מתאים יותר כאשר יש מכפלות

**Maximum A Posteriori (MAP) Estimator**

התפלגות Prior היא ההתפלגות *האמיתית* על . התפלגות Posterior מוגדרת לפי חוק בייס: . נגדיר

השוויון השני מתקיים כי לא משפיע על המקס'.

**Expectation-Maximization (EM)**

- משתנים חסרים: ב-GMM זה היה ההסתברות .

*E-*step:

*M-step*:

**K-Nearest Neighbors:**

הדגימות בהתפלגות משותפת . נרצה למצוא שממזערת את:

*האופטימלית מקיימת****Naïve Bayes***

מניחים משהו על התפלגות ה-Prior ושהדגימות ב-Data אינן תלויות. אז יתקיים ש- אז

מאפיינים: קל למימוש, מתכלל משקל של מספר רב של פיצ'רים מהתפלגויות שונות, דורש כמות קטנה של מידע.

**Winnow2**

הפעם , אפילו שהאלגוריתם עובד גם עבור . נניח שיש הפרדה בין הדוגמאות השליליות לחיוביות: אמ"מ ו- אמ"מ . נסמן .

1. נאתחל .
2. בהנתן דוגמא נעריך את .
3. ב-Flase-Negative נבצע וב-False-Positive נבצע .

**משפט:** אם קיים מפריד ונריץ את Winnow2 אז מספר השגיאות חסום ע"י:

**השוואה בין Perceptron ו-Winnow**

פרספטרון אדטיבי פשוט אבל בעייתי עבור מספר רב של מימדים ומשקלים גבוהים. הוא יכול לקבל ערכים מונוטונים בצורה טבעית. ווינוו מולטיפליקטיבי, מתמודד יפה עם ערכים ומימדים גבוהים אבל יש צורך בשינוי של האלגוריתם על מנת שיתמודד עם ערכים מונוטונים ולא דיסקרטיים. בנוסף, יש לבחור עבורו עוד משתנה תתא.

**Linear, Quadratic Programming SVD**

מטריצה , אז יש מטריצות אורתונורמליות ומטריצה אלכסונית , כך ש-. כש- הע"ע של .

Pseudo-inverse: ו-. , אז

**כופלי לגראנז'**

מחפשים: . ניתן להבחין בין מקס' למינ' לפי חיוביות ממש של ההסיאן.

**נגזרות**

**דוגמאות**

חישוב Maximum Likelihood Estimator של התפלגות נורמלית

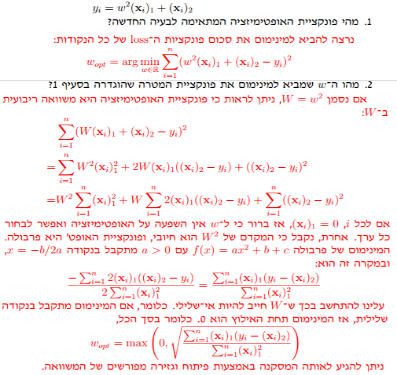
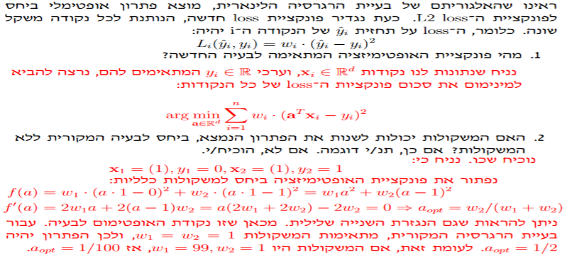
גזירה והשוואה לאפס נותנת וגם .

**נקודות ממבחנים ישנים:**

**שאלה 1:** הפרדה בין הערכים עם הסיווגים השונים: (חשוב: תלוי בצורה!!)

* + - מלבן דו-מימדי - יכול להפריד.
    - רגרסיה לוגיסטית - לא יכולה להפריד (סעיף רגרסיה לוגיסטית).
    - SVM עם Polynomial kernel ללא הגבלה של המימד- יכול להפריד לעיגול.
    - SVM עם Polynomial kernel עם הגבלה d=2- לא יכול להפריד כיוון שתיוצר פרבולה.
    - SVM עם Gaussian kernel - יכול להפריד לעיגול עם פינות מוזרות (סעיף SVM להסבר).
    - עץ החלטה עם Decision stump - יכול להפריד, תחלק את המישורים. (מלבן עם הצירים)
    - 1-Nearest Neighbor או 3 - יכול להפריד אבל תלוי במרחקים בין הנקודות !!!!!
    - AdsBoost - יכול להפריד וליצור ריבוע.
    - Perceptron - לא יכול להפריד כיוון שהוא מפריד לינארי.

**שאלה 2:**

* + - בעיית רגרסיה: 
    - 

**שאלה 2:** שאלות על טרנספורמציה:

* + - PCA - חיבור: זהה (מירכוז יבטל את ההשפעה), כפל: זהה (מתיחה אחידה לא תשנה), D: לא בהכרח (כל קורדינטה בהפרד), U: לא בהרח (סיבוב של הצירים)
    - EM - חיבור: כן כיוון שהתוחלת תשתנה גם, כפל: לא בהכרח זהה (צריך לשנות את השונות ההתחלתית), D: לא זהה, U: זהה סיבוב אחיד לא ישנה (סיבוב לתוחלת).
    - Hard margin SVM - חיבור: זהה (המפריד יזוז עם הנקודות), כפל: זהה (שינוי רק בקנה המידה), D: לא זהה (דוגמא שתשנה את המפריד), U: זהה (המפריד יסתובב באופן מתאים).

**שאלות נוספות:**   
