שיטות מתקדמות בלמידה חישובית

תוכן עניינים

בעיית הלמידה

הגדרת ה"לומד" (Learner)

$f_w: X \to Y$ פונקציה התלויה ב- w, המטרה היא למצוא את w.
הקלט לבעיה הוא $X - domain\ set$ – ווקטור באורך קבוע $x \in R^d$.
הפלט של הבעיה הוא $Y - label\ set$ – במקור $y \in \{0,1\}$ או $y \in \{-1,+1\}$ או $y \in R$.
קיימת התפלגות $D: X \times Y$ קבועה שאינה ידועה, וההנחה היא שקיימת פונקציה אמיתית $f_{w^*}(x) = y$, עבור כל $x \in X, y \in Y$. נתון סט אימון (training set) שהוא דגימה מ-D, המכיל דוגמאות מתויגות: $S = \{(x_i, y_i)\}$, כך שלכל i, $x_i \in X, y_i \in Y$ ו- $(x_i, y_i) \sim D$.
הקלט ללומד הוא X, Y וסט האימון S. הפלט הוא ההיפותזה $f_w \in F$.
הלומד נמדד ע"י אבלואציה. במסווג בינארי, משתמשים ב- error rate. מגדירים $\hat{y} = f_w(x)$ והשגיאה מוגדרת ע"י: $err = \frac{1}{m}\sum_{i=1}^{m} 1\{y_i \neq \hat{y}_i\}$, כאשר $1\{y_i \neq \hat{y}_i\}$ היא פונקציית האינדיקטור (1 אם ישנה שגיאה והתחזית שונה מהתיוג, 0 אחרת), ובסה"כ err היא השגיאה היחסית מכל הדוגמאות בסט הבדיקה. במסווגים מסוג multi-class משתמשים בכל מיני מדדים אחרים, למשל WER (Word Error Rate) ו- BLEU ב-MT, כפי שנראה בהמשך.

המטרה האולטימטיבית בכל אלגוריתם למידה היא $w^* = argmin_w\left(E_{(x,y)\in D}\left[L\left(y, f_w(x)\right)\right]\right)$ כאשר y הוא ה- true label, $f_w(x)$ הוא ה- predicated label, ו- L היא פונקציית ההערכה (Loss), המודדת את הטעות. כלומר, המטרה היא למצוא את הפרמטר עבור הלומד שימזער את תוחלת הטעויות לכל דוגמה אפשרית (x, y) מההסתברות D.

אלגוריתמי למידה

במסווג בינארי, היינו רוצים לסווג את x ל- 1+ במידה ו-$P(x|+1) > P(x|-1)$, אחרת ל- 1-.
לצורך כך, יש ללמוד את ההסתברויות מחומר האימון. כלומר, אנחנו רוצים ללמוד התפלגות D (שיתכן שהינה אינסופית) מסט דוגמאות סופי. הבעיה היא כאשר x הוא ממימד גבוה (מכיל הרבה מאפיינים - high-dimensional feature space), ולכל מאפיין ב-x יש הרבה ערכים אפשריים. במקרה זה, יש צורך בסט אימון עצום כדי שנוכל לצפות בכמה דוגמאות מכל צירוף של הערכים. עם סט אימון סופי, בעיית הדלילות מחמירה וכוח החיזוי יורד ככל שהמימד עולה – זה נקרא "קללת המימדים". לצורך כך, קיים אלגוריתם Naïve Bayes, שעושה הנחה של אי-תלות בין המימדים, וכך יש צורך לצפות רק במופעים המכילים ערכים מסוימים ולא צירופים של ערכים עבור מאפיינים שונים. הבעיה היא שלא תמיד אכן קיימת אי-תלות ולכן לא תמיד ניתן לצפות לתוצאות טובות מאלגוריתם זה.

רגולריזציה

בעיה נוספת שקיימת באלגוריתמי למידה, היא בעיה של Over-fitting: האלגוריתם לומד טוב מדי את דוגמאות האימון, אך לא נותן תחזיות נכונות לסט בדיקה נפרד. מסיבה זו, לא ניתן להחליף את התוחלת בנוסחה $w^* = argmin_w\left(E_{(x,y)\in D}\left[L\left(y, f_w(x)\right)\right]\right)$ ע"י ממוצע של דוגמאות האימון, כלומר: $w^* \neq argmin_w\left(\frac{1}{m}\sum_{i=1}^{m}\left[L\left(y_i, f_w(x_i)\right)\right]\right)$. הפתרון הוא להשתמש בפונקציית רגולריזציה: $w^* \neq argmin_w\left(\frac{1}{m}\sum_{i=1}^{m}\left[L\left(y_i, f_w(x_i)\right)\right] + R(w)\right)$ שנועדה "להתרחק" מדוגמאות האימון.
<u>VC</u> – מס' שמגדיר לכל פונקציה את יכולת הלמידה שלה. אם $V = \infty$, הפונקציה יכולה ללמוד כל דבר, ואם $V = 1$, הפונקציה יכולה ללמוד רק בעיות פשוטות כמו: x-x---x---|o—o---o.
הבעיה עם פונקציה שיכולה ללמוד כל דבר היא שהיא עושה over-fitting לדוגמאות האימון. אם x לא מופיע בדוגמאות האימון, לא ימצא y מתאים, ואם הוא כן נמצא, ה- y שיחזור יהיה בדיוק כמו בדוגמאות האימון. לכן יש להגביל את ה- VC של הפונקציה. ההגבלה היא על הפרמטרים w, למשל $\frac{1}{2}||w||^2$.

אלגוריתם פרספטרון

אלגוריתם פרספטרון (Perceptron) – הוא אלגוריתם לסיווג מבוקר שפותר את הבעיה הקלאסית של למידה מקוונת (Online) של מישורים מפרידים (hyper-planes / half-spaces). זהו סוג של מסווג ליניארי, כלומר אלגוריתם סיווג שהתחזיות שלו מתבססות על פונקציית מנבא ליניארי שהיא צירוף ליניארי של ווקטור משקלות עם ווקטור המאפיינים. האלגוריתם מאפשר למידה מקוונת, בכך שהוא מעבד את הדוגמאות באימון אחת-אחת. האלגוריתם לומד את פונקציית המשקלות ואת הסף המפריד בין ניבוי שלילי וחיובי. הרעיון הוא לבדוק את השגיאות ולעדכן את הלומד לפי הדוגמאות, וסט האימון הוא אינסופי (אין סט אימון וסט בדיקה בניגוד לאלגוריתם batch).

האלגוריתם

$w_0 = 0$
$S = \{(x_i, y_i)\}$
for $i = 1\ to\ m$
    predict $\hat{y_i} = f_{w_{i-1}}(x_i) = sgn(w_{i-1} \cdot x_i)$
    if $(\hat{y_i} \neq y_i)$
        update w: $w_i{=}w_{i-1} + y_i x_i$
    end
end

הסבר: בהינתן סט דוגמאות מתויגות $S = \{(x_i, y_i)\}$, האלגוריתם לומד את ווקטור המשקלות. מתחילים מווקטור משקלות אפס. לכל דוגמת אימון מתויגת, מנבאים את ה-label לפי ווקטור המשקלות הנוכחי. אם הניבוי נכון, לא משנים את ווקטור המשקלות. אם הוא לא נכון, משנים את w ע"י הוספה או החסרה של ווקטור המאפיינים x של הדוגמה הנוכחית (תלוי ב- label האמיתי של x).

החיזוי

$\hat{y} = f_w(x) = sgn(w \cdot x) = sgn\left(\sum_{j=1}^{d} w_j x_j\right)$ עבור $w, x \in R^d$ ו- $sgn(\pi) = \begin{cases} \pi > 0 & +1 \\ \pi < 0 & -1 \end{cases}$.

הנחות:

1. אנחנו נתייחס לסף כאל 0, כלומר במקום להשתמש ב- $wx + b$ נשתמש ב- wx. הסף יכול להיות כל מס' אחר. כדי לקבוע סף b, נוסיף מאפיין נוסף ונגדיר $x^{new} = (x, 1)$. הלומד ילמד את המשקל של המאפיין הנוסף שהוא b.
2. ניתן להניח שדוגמאות האימון מנורמלות, כלומר לכל i, $\left||x_i|\right| = 1$. אם לא, ניתן לנרמל אותם מכיוון שזה לא משפיע על החיזוי: $sign(wx) = sign\left(\frac{(wx)}{||x||}\right)$.

הפרדה

הפרדה (Separability) – קיימת פונקציה $l: X \to Y$ המקיימת $y_i = l(x_i)$. זו תכונה של סט האימון.
הפרדה ליניארית - $l(x_i) = sign(wx_i + b)$.

שוליים (margin) - $\gamma = \min_i \frac{|w^* \cdot x_i|}{||x_i||} = \min_i |w \cdot x_i|$. מוגדר להיות המרחק המינימלי של דוגמת אימון מהמישור המפריד $w^* x = 0$. זו הרובסטיות של האלגוריתמים. הרעשה (perturbation) של x_i לא תשנה את הסיווג.

<u>משפט 1</u> – יהא $S=\{(x_i,y_i)\}$ סט דוגמאות אימון מופרד ליניארי (קונסיסטנטי עם ווקטור המשקלות המנורמל w^*, כלומר, קיימת פונקציה f_{w^*} שעבורה לכל (x_i,y_i) מדוגמאות האימון $f_{w^*}(x_i)=y_i$), אז הפרספטרון יעשה M עדכונים (שגיאות), עד שיתכנס ל-$\widetilde{w}$ שעבורו אין שגיאות לדוגמאות האימון ($y_i=f_{\widetilde{w}}(x_i)$), ומתקיים $M\le\frac{1}{\gamma^2}$ תחת ההנחות הנ"ל.

<u>אינטואיציה</u> - למשל, ב- R^2, אם קיים ישר שהוא מפריד ליניארי של כל דוגמאות האימון, אז האלגוריתם ימצא ישר כלשהו שהוא גם מפריד ליניארי שלהן (לא בהכרח אותו הישר).

<u>הוכחה</u> – מגדירים פונקציית פוטנציאל $\phi(i)=w_i\cdot w^*$. בהתחלה $\phi(0)=0$ כי $w_0=0$. אם $\phi(i)=||w_i||$ זה אומר ש- $w_i=w^*$ והפרספטרון חוזה את כל הנקודות x_i נכונה.

1. $w_i\cdot w^*=(w_{i-1}+y_ix_i)\cdot w^*=w_{i-1}\cdot w^*+y_iw^*x_i\ge w_{i-1}\cdot w^*+\gamma$
כלומר, בכל טעות, פונקציית הפוטנציאל גדלה לפחות ב- γ. זה נובע מהגדרת השוליים ($\gamma\le|w\cdot x_i|$), מכך ש- $y_i=\pm1$ ומכך שהגודל $y_iw^*x_i$ תמיד חיובי כי הסימן של y_i ושל w^*x_i זהה.
2. $||w_i||^2=||w_{i-1}+y_ix_i||^2=||w_{i-1}||^2+2y_iw_{i-1}x_i+||x_i||^2\le||w_{i-1}||^2+1$
נובע מהנחה 2 ($||x_i||^2=1$) ומכך שזו טעות (ממשיכים לעדכן) ולכן $y_iw_{i-1}x_i<0$.

מ-1, $w_M\cdot w^*\ge w_{M-1}\cdot w^*+\gamma\ge\cdots\ge w_0\cdot w^*+M\gamma\ge M\gamma$.
מ-2, $||w_M||^2\le||w_{M-1}||^2+1\le\cdots\le||w_0||^2+M\le M$.
משני אלה ביחד, $M\gamma\le w_M\cdot w^*=||w_M||\le\sqrt{M}\Leftarrow$ $M\le\frac{1}{\gamma^2}$. ■

אם הנתונים מופרדים היטב (כלומר γ גדול), אז זמן הריצה יהיה טוב למדי ולכן כדאי להשתמש בפרספטרון במקרה זה. יתכן ש-S אינו מופרד לגמרי, כלומר קיים w^* שעבורו כמעט כל דוגמאות האימון מתויגות נכון. גם במקרה הזה, אלגוריתם פרספטרון מתכנס, אך מס' השגיאות שהוא עושה גדול יותר.

<u>משפט 2</u> – אם S אינו מופרד ליניארית, אז מס' השגיאות של פרספטרון הוא $M\le\frac{1}{\gamma^2}+\frac{2}{\gamma}\cdot TD_\gamma$ כאשר TD_γ הוא הסכום המינימלי הדרוש להזזת הנקודות במדגם הנמצאות בצד הלא נכון של המפריד ולהוסיף להן שוליים γ.

<u>הוכחה</u> –

1. $w_M\cdot w^*\ge M\gamma-TD_\gamma$
פונקציית הפוטנציאל גדלה לפחות ב-γ עבור הדוגמאות המופרדות ע"י w^*. עבור כל דוגמה אחרת מתקיים ש- $y_iw^*x_i=-1$ ולכן מכפילים את הערך הזה בסכום המינימלי הנדרש להזזת הנקודות כך ש-w^* יפריד בין כל הנקודות.
2. נשאר כמו במשפט 1: $||w_M||^2\le M$

משני אלה ביחד, $M\gamma-TD_\gamma\le w_M\cdot w^*=||w_M||\le\sqrt{M}\Leftarrow$
ע"י העלאה בריבוע: $M^2\gamma^2-2M\gamma TD_\gamma+TD_\gamma^2\le M\Leftarrow$
$TD_\gamma^2>0$ ולכן: $M^2\gamma^2-2M\gamma TD_\gamma\le M\Leftarrow M\gamma^2-2\gamma TD_\gamma\le1\Leftarrow M\le\frac{1}{\gamma^2}+\frac{2}{\gamma}TD_\gamma$. ■

<u>משפט 3</u> – עבור מדגם S שאינו מופרד ליניארית, $||x_i|| \leq 1$, נניח וקטור כלשהו w שעבורו $||w|| = 1$ ומספר שוליים כלשהו $\gamma > 0$. נגדיר $d_i = \max\{0,1 - y_i w x_i\}$ ו- $D = \sqrt{\sum_{i=1}^{m} d_i^2}$ אז $M \leq \left(\frac{D+1}{\gamma}\right)^2$.

<u>אלגוריתם פרספטרון כבעיית ספיקות</u>

ניתן לחשוב על אלגוריתם פרספטרון כאל סט אילוצים: כל דוגמה בסט האימון היא אילוץ: $y_i w_i x_i > 0$, כלומר שהניבוי של w זהה ל-label. סט האילוצים הוא: $\forall i \in \{1,\dots,m\}: y_i w x_i > 0$ ויש למצוא w המקיים את כל האילוצים. בפתרון של בעיית הספיקות (feasibility), מתקבל בדיוק אלגוריתם פרספטרון. האלגוריתם פותר בעיה ספרבילית (בעיה שקיים לה מפריד ליניארי), אבל גם בעיות לא ספרביליות (לא ראינו למה).

ניתן להפוך את בעיית הספיקות לבעיית אופטימיזציה כך שכל דוגמת אימון תהיה רחוקה מהמישור המפריד ע"י שוליים γ.

האלגוריתם המתוקן: (הנחה: הווקטורים x_i מנורמלים).

$w_0 = 0$
$S = \{(x_i, y_i)\}$
for $i = 1$ to m

$$\text{predict } \hat{y}_i = f_{w_{i-1}}(x_i) = \begin{cases} +1 & \frac{w_{i-1}x_i}{||w_{i-1}||} \geq \frac{\gamma}{2} \\ -1 & \frac{w_{i-1}x_i}{||w_{i-1}||} \leq -\frac{\gamma}{2} \end{cases}$$

if $\left(-\frac{\gamma}{2} \leq \frac{w_{i-1}x_i}{||w_{i-1}||} < \frac{\gamma}{2}\right)$
update w: $w_i = w_{i-1} + y_i x_i$
end
end

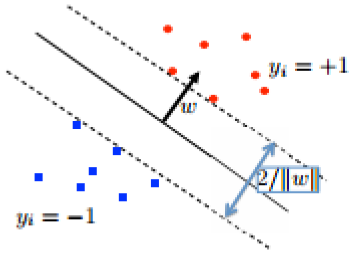
אם $y_i w_{i-1} x_i < \frac{\gamma}{2}$ אם $y_i w_{i-1} x_i < 0$ (הנקודה נמצאת מהצד הלא נכון של המישור המפריד) או אם $w_{i-1} x_i \in \left[0, \frac{\gamma}{2}\right]$ (הנקודה נמצאת בתוך השוליים). כלומר, אם האלגוריתם מתכנס, מתקבל ווקטור משקלות w כך שכל נק' מדגם תהיה רחוקה לפחות $\frac{\gamma}{2}$ מהמישור המפריד $w^* x$ ונקבל שוליים γ.

היתרון על פני Perceptron קלאסי הוא בכך שזה נותן יותר רובסטיות: הרעשה (perturbation) של x_i לא תשנה את הסיווג שלו כי הוא מספיק רחוק מהמישור המפריד.

נותר רק להראות שגם האלגוריתם המתוקן מתכנס. ניתן להוכיח זו באופן דומה למשפט לפיו אם S מופרד ליניארית, אז הפרספטרון יעשה M שגיאות עבור $M \leq \frac{1}{\gamma^2}$, כאשר $\gamma = \min_i |w \cdot x_i|$ המרחק של דוגמת האימון הקרובה ביותר למישור המפריד. באלגוריתם המתוקן יהיו יותר טעויות, אך גם הוא מתכנס (ההוכחה המלאה נמצאת במאמר הראשון).

SVM

SVM (Support Vector Machine) הוא מודל סיווג מבוקר ליניארי. בדומה לאלגוריתם פרספטרון, הוא מוצא מישור מפריד, אך הוא אינו מקוון. בנוסף, במודל SVM, דוגמאות האימון הן נקודות (x, y) במישור, והמטרה היא למצוא מישור מפריד כך שבין דוגמאות האימון לבין המפריד יהיו שוליים רחבים ככל האפשר (Support Vectors). המרחק הזה הוא השוליים (margin). כאשר לשוליים בצד האחד נמצאות הדוגמאות החיוביות ובצד השני הדוגמאות השליליות. החיזוי מתבצע ע"י $sign(wx)$.



באופן פורמלי:

$$w^* = argmin_w \left(\frac{1}{2}||w||^2\right), s.t.\ \forall i\ y_i(w \cdot x_i) \geq 1$$

כלומר, רוצים למצוא את ווקטור המשקלות w^* המייצר שוליים $margin = \frac{1}{2}||w||^2$, כך שהדוגמאות מתויגות נכון ($y_i(w \cdot x_i) \geq 0$) ואינן בתוך השוליים (לא מתקיים $0 < y_i(w \cdot x_i) < \frac{1}{2}||w||^2$).

<u>משמעות השוליים</u>

אם דוגמאות האימון מופרדות ליניארית, ניתן לבחור את המישורים המפרידים כך שייווצרו ביניהם שוליים (margin) המפרידים בין דוגמאות האימון, ולנסות להרחיב את השוליים ככל הניתן.
בניגוד למישור היחיד המוגדר ע"י פרספטרון ($w \cdot x = 0$), ב-SVM המישורים מוגדרים ע"י $w \cdot x = 1$ ו- $w \cdot x = -1$. דוגמאות אימון שנמצאות מעבר למפריד הראשון הן דוגמאות חיוביות, ודוגמאות אימון שנמצאות מעבר למפריד השני הן דוגמאות שליליות.

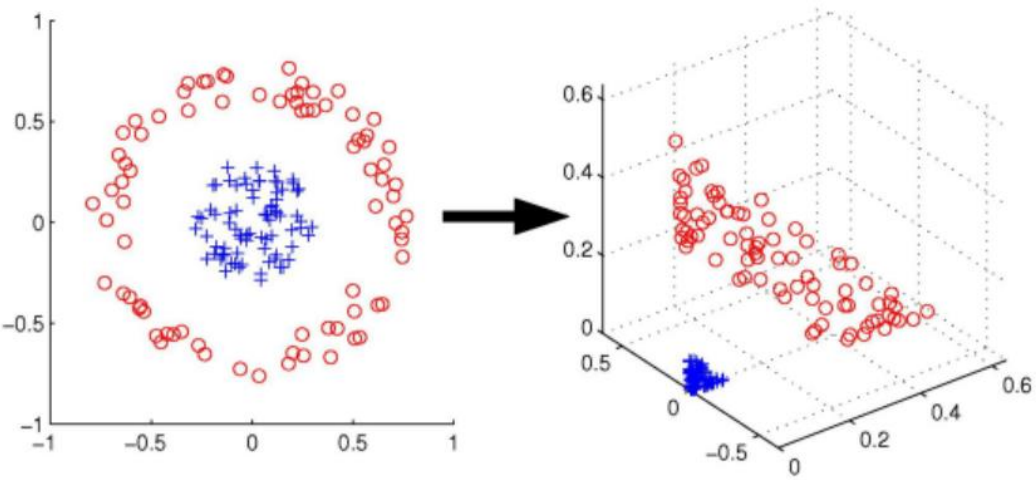
ע"י שימוש בגיאומטריה, ניתן להבחין שהמרחק בין שני המישורים המקבילים הוא $\frac{2}{||w||} = \frac{|1--1|}{||w||}$.
לכן, על מנת למקסם את השוליים יש להביא למינימום את $||w||$. בפועל, מביאים למינימום את הביטוי $\frac{1}{2}||w||^2$ כדי שיהיה קל לגזור אותו (לאחר הגזירה נקבל $||w||$). זה לא משנה את התוצאה.

נשים לב שהאילוצים נועדו לכך שדוגמאות אימון לא ייפלו בין השוליים. לכן לכל דוגמאות אימון, או שמתקיים $w \cdot x_i \geq 1$ והדוגמה חיובית $y_i = 1$, ואז $y_i(w \cdot x_i) \geq 1$, או שמתקיים $w \cdot x_i \leq -1$ והדוגמה שלילית $y_i = -1$, ואז גם יתקיים $y_i(w \cdot x_i) \geq 1$.

מהי המשמעות של השוליים? כלומר, מה יקרה אם נגדיר את האילוצים כך: $y_i(w \cdot x_i) \geq 7$?
אין משמעות. המרחק הפונקציונלי של דוגמת אימון מהמישור הוא $y_i(w \cdot x_i)$. אם הדוגמה תויגה נכון ו-$|wx_i|$ גדול, המרחק יהיה מס' חיובי גדול. מרחק פונקציונלי של דוגמת אימון מייצג את הביטחון בסיווג שלה. לכן, נראה שאם נדרוש $y_i(w \cdot x_i) \geq 7$, נגדיל את השוליים הפונקציונליים. הבעיה היא שלאלגוריתם יש מאפיין שהופך את המרחק הפונקציונלי ללא רלוונטי. האלגוריתם ילמד את ווקטור המשקלות $7w$, המקיים את כל האילוצים ($y_i(w \cdot x_i) \geq 1 \Longrightarrow y_i(7w \cdot x_i) \geq 7$). מכיוון שהסיווג מתבצע על סמך הסימן של $w \cdot x_i$ ולא מתחשב בגודל שלו, כל x יתויג ע"י $7w$ באופן זהה ל-w.

טריק Kernel

גם אם דוגמאות האימון אינן מופרדות לינארית, לפעמים עדיין אפשר למצוא מפריד ע"י SVM.
מעבירים את הדוגמאות x_i ואת ווקטור המשקלות w למימד גבוה יותר ע"י פונקציית מיפוי
$\phi: R^{d_1} \rightarrow R^{d_2}$ כאשר $d_1 < d_2$, מתוך הנחה שבמימד גבוה יותר תהיה הפרדה ליניארית.



לדוגמה: הנתונים שבגרף השמאלי אינם מופרדים ליניארית במימד 2. ע"י העברתם למימד 3, ע"י פונקציית המיפוי: $\varphi: R^2 \rightarrow R^3$ המוגדר באופן הבא: $\varphi(x_1, x_2) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$ מקבלים את הגרף מימין שהוא מופרד ליניארית.
המפריד יהיה מהצורה $wx = 0 \Rightarrow\ w_1x_1^2 + w_2\sqrt{2}x_1x_2 + w_3x_2^2 = 0$.

חישוב פונקציית המיפוי יכול ליצור בעיה חישובית בגלל המימד הגבוה d_2. עם זאת, האלגוריתמים המוצאים מפריד ליניארי, משתמשים רק במכפלות הסקלריות של דוגמאות האימון x_i, x_j (זה נובע מהגדרת הבעיה הדואלית, שלא נתעמק בה). לכן, ניתן "לזרוק" את כל הנתונים ולהשתמש אך ורק במטריצת Gram שהיא המטריצה של כל המכפלות הסקלריות של הדוגמאות: XX^T.
כאשר משתמשים בפונקציית מיפוי φ, נצטרך להשתמש במטריצה שמכילה את המכפלות הסקלריות על המיפויים, כלומר, המקיימת $K\left(x_i, x_j\right) =\ \varphi(x_i) \cdot \varphi(x_j)$. מסתבר שניתן להשתמש בפונקציית Kernel, פונקציה $K: R^{d_1} \times R^{d_1} \rightarrow R$, המחשבת dot-product בין שני ווקטורים ומקיימת: $K(x, y) =\ \varphi(x) \cdot \varphi(y)$. בצורה זו, המיפוי φ מתבצע באופן לא מפורש. יש לחשב את המטריצה פעם אחת, מבלי הצורך להכיר את φ, וניתן לקבל באופן ישיר את הערך $\varphi(x_i) \cdot \varphi(x_j)$ מ-$K\left(x_i, x_j\right)$.

פונקציית Kernel לדוגמה: $K(x, y) = (1 + x \cdot y)^d$.

<u>SVM כבעיית אופטימיזציה</u>

SVM היא בעיית אופטימיזציה עם אילוצים על דוגמאות האימון ופונקציית מטרה שתלויה ב- w.

$$w^* = argmin_w\left(\frac{1}{2}||w||^2\right), s.t.\ \forall i\ w(y_i \cdot x_i) \geq 1$$

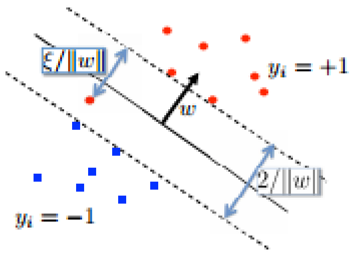
רוצים למצוא את השוליים המקסימליים, כלומר למקסם את $\frac{2}{||w||}$. האילוצים דואגים לכך שכל נק' מדגם תהיה רחוקה לפחות 1 מהמישור המפריד $(y_i w^* x_i \geq 1)$. המרחק של x מהמישור המפריד מחושב כהטלה[1] של x על w^* (מרחק גיאומטרי $wx = \frac{|wx|}{||w||}$) ולא כנורמה (מרחק אוקלידי).

ההגדרה הפורמלית של SVM כבעיית אופטימיזציה: $\min\frac{1}{2}||w||^2, s.t.\ 1 - y_i w x_i \leq 0$.

קיימת גרסה פחות קשיחה של SVM (Soft Margin) שבה מאפשרים לחלק מנקודות המדגם להיות בצד הלא נכון של המדגם, ואז "משלמים" על העברת הנקודות, ואנחנו רוצים למזער את התשלומים האלה:

$$\min\left[\frac{1}{2}||w||^2 + C\sum_i \xi_i\right]$$
$$s.t.\ 1 - \xi_i - y_i w x_i \leq 0, \xi_i \geq 0$$

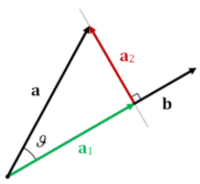
כאשר C הוא פרמטר שהמשתמש נותן והוא קובע כמה רוצים למזער את ה- margin וכמה להתייחס לטעויות. אם C קטן, יותר חשוב ה- margin מאשר למזער את הטעויות ($\frac{1}{2}||w||^2$ תורם לפונקציית המטרה יותר מאשר $C\sum_i \xi_i$), ואם C גדול, חשוב למזער את הטעויות יותר מאשר את ה- margin.
ξ_i הוא המחיר לטעות עבור הנקודה (x_i, y_i), והוא כמובן חיובי.



אנחנו לא נעמיק ב-Soft Margin.

---

[1] <u>הטלה (Projection)</u> – הטלה של ווקטור $\vec{a}$ על ווקטור $\vec{b}$ מוגדרת ע"י $a^{proj} = \frac{\vec{a}\cdot\vec{b}}{||\vec{b}||}$. זהו מס' וזה האורך של הניצב.



<u>הגדרת ה-Loss</u>

נניח שהדוגמאות מגיעות מהתפלגות: $(x,y)\sim\rho$. נגדיר פונקציית הפסד (Loss) כמדד לכישלון בסיווג על דוגמה מתויגת, למשל: $L(y,\hat{y})=1[y\neq\hat{y}]$, פונקציית אינדיקטור המחזירה 1 אם יש טעות בסיווג, אחרת 0.

נרצה למצוא את ווקטור המשקלות שממזער את תוחלת ההפסד על הנתונים:

$$w^*=argmin_w\left(E_{(x,y)\in\rho}\left[L\left(y,f_w(x)\right)\right]\right)$$

ההתפלגות האמיתית ρ אינה ידועה, אלא נתון רק סט דוגמאות אימון $S=\{(x_i,y_i)\}$. לא ניתן להחליף את התוחלת בממוצע על דוגמאות האימון, כי זה גורם ל- Over-fitting לסט האימון:

$$w^*\neq argmin_w\left(\frac{1}{m}\sum_{i=1}^{m}\left[L\left(y_i,f_w(x_i)\right)\right]\right)$$

הפתרון הוא להשתמש בפונקציית רגולריזציה שהיא הגבלה על w:

$$w^*\neq argmin_w\frac{1}{m}\sum_{i=1}^{m}\left[L\left(y_i,f_w(x_i)\right)\right]+\frac{\lambda}{2}\left||w|\right|^2$$

הוספת הרגולריזציה לפונקציית המטרה דואגת לכך שהפונקציה לא תהיה בעלת VC גבוה (פונקציה ממימד גבוה שנצמדת לנקודות המדגם), ע"י גורם הרגולריזציה: $\frac{1}{2}\left||w|\right|^2$. λ הוא קבוע הרגולריזציה.

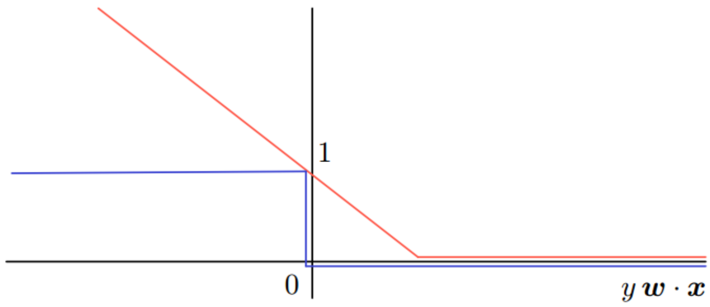
בעיה נוספת היא שלא תמיד ניתן לגזור את L ולא תמיד יש לה מינימום יחיד, לכן עושים לה קירוב:

$$w^*\neq argmin_w\frac{1}{m}\sum_{i=1}^{m}\left[l(w;x_i,y_i)\right]+\frac{\lambda}{2}\left||w|\right|^2$$

L נקרא task loss (או 0-1 Loss) והוא מדד לכישלון. במקום להביא למינימום את L, ניתן להביא למינימום את החסם העליון שלו, שהוא קמור[2] (ולכן יש לו מינימום).
זוהי הפונקציה l הנקראת hinge loss.

$$L(y,\hat{y})=1[y\neq\hat{y}]=1[ywx\leq 0]\leq\max\{0,1-ywx\}=l(w;x,y)$$

ניתן להסביר את המעבר האחרון בצורה גרפית. הגרף הבא מציג את L ואת l כפונקציה של ywx:



---

[2] $f(x)$ קמורה (convex) אם לכל $x_1,x_2\in R^d$, $\lambda\in[0,1]$, $f(\lambda x_1+(1-\lambda)x_2)\leq\lambda f(x_1)+(1-\lambda)f(x_2)$
כאשר הפונקציה קמורה, ניתן לעשות לה אופטימיזציה כי יש לה מינימום יחיד. אם הפונקציה קעורה (Concave), אפשר לעשות אופטימיזציה למינוס הפונקציה.

L היא פונקציית מדרגה: כאשר $ywx > 0$ ערכה 0 וכאשר $ywx \leq 0$ ערכה 1.
l, עבור $ywx > 1$ ערכה 0, ולפני כן ערכה $1 - ywx$, פונקציה לינארית יורדת עם נק' חיתוך עם ציר ה- y ב-1. זו פונקציה קמורה, וניתן לראות בבירור שהיא חסם עליון ל-L, כלומר, אם נמזער אותה, נמזער גם את L.

לסיכום, הנוסחה המעודכנת ל-SVM היא:

$$w^* = argmin_w \left(\frac{1}{m} \sum_{i=1}^{m} [\max\{0, 1 - ywx\}] \right) + \frac{\lambda}{2} \left| |w| \right|^2$$

כאשר λ הוא פרמטר שקובע כמה רגולריזציה וכמה Loss. הבעיה היא שהפונקציה הזו היא לא לגמרי גזירה (היא לא גזירה בנקודה 1) אלא רק "כמעט גזירה". בהמשך נראה דרך להתמודד עם זה.

<u>Stochastic Gradient Descent לפתרון SVM</u>

<u>גרדיאנט – הגדרות כלליות</u>

<u>תת גרדיאנט (Sub-gradient)</u> – וקטור λ הוא תת-גרדיאנט של הפונקציה $f: \mathbb{R}^d \to \mathbb{R}$ בנקודה w (ווקטור) אם לכל $u \in \mathbb{R}^d$: $f(u) - f(w) \geq (\bar{u} - \bar{w})^T \cdot \bar{\lambda}$.
זוהי הכללה של נגזרת לפונקציות שהן לא לגמרי גזירות. לדוגמה, הפונקציה hinge-loss היא גזירה חוץ מאשר בנקודה 1. בגרדיאנט מתקיים שוויון $f(u) - f(w) = (\bar{u} - \bar{w})^T \cdot \bar{\lambda}$ וקיים בדיוק λ אחד המקיים את התנאי. בדר"כ לא נשתמש בהגדרה הישירה אלא בחלוקה למקרים (בהמשך).

<u>סט דיפרנציאלי</u> - $\partial f(w)$ הוא סט כל התת-גרדיאנטים בווקטור w. התכונות:

1. אם הסט הדיפרנציאלי מכיל נקודה אחת, אז $\nabla_w f(w) = \partial f(w)$.
2. אם $f(w) = \max_{i=1,\dots,r} f_i(w)$ ו- $f_i(w)$ הן פונקציות גזירות ומתקיים $j = argmax_j f_j(w)$, אז הגרדיאנט של f_j ב-u הוא תת-גרדיאנט של f ב- u. במילים אחרות, התת-גרדיאנט של f הוא הגרדיאנט של הפונקציה הממקסמת f_j.

ההגדרה של גרדיאנט נחוצה לנו לצורך פתרון בעיית אופטימיזציה של פונקציה שאינה גזירה. בפרט, לבעיה מסוג $w^* = argmin_w\big(F(w)\big)$ כאשר $F(w) = \frac{1}{m}\sum_{i=1}^{m} f_i(w)$, ישנו אלגוריתם כללי לפתרון:

$w_0 = 0$
for $t = 1$ to T
    choose (x_i, y_i) uniformly
    $w \leftarrow w - \eta \partial f_i(w)$
end

אלגוריתם זה נקרא Stochastic Sub-gradient Descent (SSGD). האלגוריתם עובד גם עבור פונקציה שאינה קמורה, ובמקרה הזה הוא ימצא נקודת קיצון מקומית.

<u>אלגוריתם SGD לפתרון SVM</u>

ניתן להפעיל את אלגוריתם SSGD על פונקציית המטרה של SVM, ללא האילוצים:

$$w^* = argmin_w\left(\frac{1}{m}\sum_{i=1}^{m}[\max\{0,1-ywx\}]\right)+\frac{\lambda}{2}||w||^2$$

נפתח את התת-גרדיאנט של $\max\{0,1-ywx\}$:

מסעיף 2 של הסט הדיפרנציאלי, נקבל כי: $\partial_w[\max\{0,1-ywx\}] = \begin{cases}-y_ix_i & 0<1-y_iwx_i \\ 0 & else\end{cases}$.

$$\partial_w f(w) = \partial_w\left(\frac{1}{m}\sum_{i=1}^{m}[\max\{0,1-ywx\}]\right)+\partial_w\left(\frac{\lambda}{2}||w||^2\right) = \begin{cases}-y_ix_i + \lambda w & 0<1-y_iwx_i \\ \lambda w & else\end{cases}$$

```
Input: training set S = {(x_i, y_i)}, parameter λ, η_0
Initialize: w_0 = 0
for t = 1 to T
        choose (x_i, y_i) uniformly at random
        if (1 − y_i w x_i ≥ 0)
                w_t ← w_{t−1} − η_t[λw − y_i x_i] = w_{t−1}(1 − η_t λ) + η_t y_i x_i
        else
                w_t ← w_{t−1} − η_t[λw + 0] = w_{t−1}(1 − η_t λ)
Output: w = ∑_{t=1}^{T} w_t
```

η נקרא קבוע הלמידה. להתכנסות, צריך לבחור $\eta=\frac{1}{\lambda\sqrt{t}}$ או $\eta=\frac{1}{\lambda t}$ לפי $0<1-\eta\lambda<1$. התנאי הזה בעייתי כי ב-SVM אין דרישה כזו ויכול להיות שלא יהיה פתרון. במקרה כזה, צריך לעבור לפתרון הדואלי. בסיום האלגוריתם, מחזירים את הסכום של כל ווקטורי המשקלות שנלמדו (ניתן להשמיט את הווקטורים הראשונים, לפני ההתכנסות). זה שקול לממוצע, כי ניתן לעשות Scaling לווקטור המשקלות מבלי לשנות את הסיווג.

<u>ליפשיץ ρ</u> - $|f(u)-f(w)|\le\rho||u-w||$, התת-גרדיאנט (והגרדיאנט) חסומים.

<u>משפט</u> - $E[F(\bar{w})]\le\min_{w^*}F(w^*)+\rho U\sqrt{\frac{2}{T}}$, כאשר w^* הוא הפתרון האמיתי, T הוא מס' הצעדים, $\bar{w}=\frac{1}{T}\sum_{t=1}^{T}w_t$ הוא ממוצע ווקטורי המשקלות באיטרציות השונות, $U\le\max\{||\bar{w}-w^*||\}$ המרחק בין הפתרון המשוערך לאמיתי. (ללא הוכחה).

<u>התכנסות של האלגוריתם</u> - מתי $\rho U\sqrt{\frac{2}{T}}<\epsilon$, כלומר מתי נגיע לדיוק של אפסילון בפתרון?

$$\sqrt{\frac{2}{T}}<\frac{\epsilon}{\rho U}\Rightarrow T>\frac{2\rho^2U^2}{\epsilon^2}$$

בפועל, אם יש מספיק דוגמאות (אלפים), עושים עליהן shuffle ובוחרים כל דוגמה פעם אחת. תלוי בדיוק הרצוי. האלגוריתם הזה מתכנס מהר יותר מהאלגוריתמים כמו SVM Light שהם ריבועיים.

<u>השוואה בין SVM לפרספטרון</u>

1. בפרספטרון אין שוליים (margin), ולכן האלגוריתם בודק שגיאות $y_iwx_i<0$, וב-SVM הוא בודק שגיאות margin: $1-y_iwx_i\ge0$.
2. בפרספטרון אין גורם רגולריזציה ולכן הגורם $\eta_t\lambda w$ אינו מופיע בו. זה שקול לקביעה של $\lambda=0$.
3. קבוע הלמידה בפרספטרון הוא קבוע $\eta_t=0$.

Logistic Regression

אלגוריתם נוסף לסיווג בינארי. במסווג בייסיאני, רוצים לסווג כל דוגמה למחלקה שיותר סביר שאליה היא משתייכת, כלומר $P(Y=+1|X=x) > P(Y=-1|X=x)$ יתן ערך $y=1$, אחרת $y=-1$.
הבעיה עם שיטת הסיווג הזו היא "קללת המימדים", לא ניתן להכליל אותה להרבה מאפיינים ואין לה רגולריזציה. הפתרון הוא שימוש ברגרסיה ליניארית:

$$\hat{P}(Y=+1|X=x) = \frac{1}{1+e^{-w\cdot x}} = f(x,w)$$

עדיין מקבלים ערך מספרי דומה להסתברות, אבל מקבלים אותו מפונקציה דומה למסווג ליניארי.
הנוסחה נובעת משערוך ליניארי. רוצים למצוא w שמחזיר את המחלקה הנכונה לפי log-likelihood, כלומר:

$$\log\left(\frac{p(x)}{1-p(x)}\right) = w\cdot x$$

אם $p(x) > 1-p(x)$, אז $wx > 0$, אחרת $wx < 0$.

ע"י פיתוח הנוסחה, מקבלים:

$$P(x) = \frac{e^{wx}}{e^{wx}+1} = \frac{1}{1+e^{-w\cdot x}}$$

בשלב האימון, כמו בסיווג הבייסיאני, מוצאים את w ע"י MLE (Maximum Likelihood):

$$max \sum_{i=1}^{m} \log\left(\hat{P}(Y=y_i|X=x_i,w)\right) - \frac{\lambda}{2}\left||w|\right|^2 =$$
$$max \sum_{i=1}^{m} y_i \log\left(\frac{1}{1+e^{-w\cdot x_i}}\right) - \frac{\lambda}{2}\left||w|\right|^2 = min \sum_{i=1}^{m} y_i \log(1+e^{-w\cdot x_i}) + \frac{\lambda}{2}\left||w|\right|^2$$

המטרה היא למקסם את ה-likelihood לפי המודל עבור כל דוגמאות האימון, ומוסיפים רגולריזציה.

גוזרים על מנת למצוא את w (GD):

$$[\nabla_w]_i = \frac{-x_i\cdot e^{-w\cdot x_i}}{1+e^{-w\cdot x_i}} y_i + \lambda w = -x_i(y_i - P_w[y_i|x_i])$$

כלומר, מתקבלת "תוחלת" של דוגמאות האימון. הפונקציה מחזירה ערך דומה להסתברות.

אלגוריתמים מקוונים ולא-מקוונים
אלגוריתמי למידה יכולים להיות מקוונים (online) או לא מקוונים (batch).

באלגוריתם batch מאמנים את המודל על סט אימון, ומשתמשים בו (inference) על סט בדיקה. כדי לבדוק את האלגוריתם, יש צורך ב- training set ו- test set. לפעמים צריך גם development set או validation set שהוא חלק שמפרידים (held out) מה- training ועליו מכיילים את הפרמטרים.

באלגוריתם online, יש לנו סט אימון המשכי (אינסופי) ואין סט בדיקה. מקבלים דוגמה, מנבאים אותה לפי הפרמטרים הנוכחיים, והשגיאות נבדקות על הדוגמה ולא על סט בדיקה. במקרה של שגיאה, מעדכנים את הפרמטרים. אם יש דוגמה שאינה מתויגת, לא עושים כלום. לדוגמה, בסינון ספאם. אם תוכנת המייל ניבאה שהמייל הנוכחי הוא ספאם ושמה לי אותו בתיקיית ספאם, אם אני אמחק את המייל הזה – זה אומר שהניבוי היה נכון, ואם אעביר אותו בחזרה לתיבת הדואר הנכנס, הניבוי היה טעות ולכן הוא יעדכן את הפרמטרים. בחיפוש, זה יעבוד עם הלינק שהמשתמש לחץ עליו. אם המשתמש לחץ על לינק שמופיע בראש רשימת המסמכים שחזרו, ה- loss יהיה נמוך, ואם הוא ילחץ על אחד מהלינקים שמופיעים נמוך ברשימה, ה- loss יהיה גבוה.

סכימת אלגוריתם Online:

for t=1,2,...,T
    get x_t
    predict $\hat{y}_t$ based on w_t
    get y_t
    compute error $(y_t, \hat{y}_t)$
    update w_{t+1}

העדכון לווקטור המשקלות הוא מהצורה: $w_{t+1} = argmin_w L$
כאשר $L = \Omega(w_{t+1}, w_t) + \eta_t l(w_{t+1}, x_t, y_t)$. η_t הוא קבוע הלמידה, שקובע כמה משקל לתת לדוגמה הנוכחית, ו- $\Omega(w, w_t)$ הוא קבוע הרגולריזציה. זו כמו בעיית אופטימיזציה על דוגמה אחת. המשמעות של קבוע הרגולריזציה במקרה הזה היא שונה ממה שראינו עד כה, ומטרתה לגרום ל-w_{t+1} לא להיות שונה מדי מ-w_t שכבר למדנו.

פרספטרון הוא אלגוריתם מקוון. נראה שמתקיימים בו הפרמטרים הבאים:

$$\Omega(w_{t+1}, w_t) = \frac{1}{2}\left|\left|w_{t+1} - w_t\right|\right|^2$$

$$l(w, x_t, y_t) = -y_t w_{t+1} x_t$$

$$\eta_t = 1$$

כדי למצוא את המינימום, גוזרים לפי $w = w_{t+1}$ ומשווים ל-0:

$$\nabla_{w_{t+1}} L = 0 \Rightarrow \nabla_{w_{t+1}} \left[\frac{1}{2}\left|\left|w_{t+1} - w_t\right|\right|^2 - y_t w_{t+1} x_t\right] = 0 \Rightarrow (w_{t+1} - w_t) - y_t x_t = 0$$
$$\Rightarrow w_{t+1} = w_t + y_t x_t$$

קיבלנו בדיוק את העדכון של פרספטרון.

באלגוריתמים אחרים, אם לא ניתן לגזור ולהשוות ל-0 (למשל כשיש ממוצע על ה- loss), משתמשים בתת-גרדיאנט (SGD).

ב- Passive-Aggressive (סוג של SVM שנקרא כך בגלל העדכון האגרסיבי של w והפסיבי של C, שהושמט כאן), מתקיים:

$$\Omega(w_{t+1}, w_t) = \frac{1}{2}\left|\left|w_{t+1} - w_t\right|\right|^2$$

$$l(w, x_t, y_t) = 1 - y_t w_{t+1} x_t$$

$$\eta_t = \tau_t$$

ובסה"כ:

$$L = \frac{1}{2}\left|\left|w - w_t\right|\right|^2 + \tau(1 - y_t w x_t)$$

על מנת למצוא את המינימום, פותרים SGD:

$$\nabla_w L = 0 \Longrightarrow (w - w_t) - \tau y_t x_t = 0 \Longrightarrow w = w_t + \tau y_t x_t$$

נציב את w כדי למצוא את τ:

$$L = \frac{1}{2}\left|\left|w_t + \tau y_t x_t - w_t\right|\right|^2 + \tau(1 - y_t(w_t + \tau y_t x_t)x_t) =$$
$$=_{(y^2=1)} \frac{1}{2}\tau^2\left|\left|x_t\right|\right|^2 + \tau\left(1 - y_t w_t x_t - \tau\left|\left|x_t\right|\right|^2\right) =$$
$$=_{(y^2=1)} \frac{1}{2}\tau^2\left|\left|x_t\right|\right|^2 - \tau^2\left|\left|x_t\right|\right|^2 + \tau(1 - y_t w_t x_t) =$$

$$= -\frac{1}{2}\tau^2\left|\left|x_t\right|\right|^2 + \tau[1 - y_t w x_t]$$
$$\nabla_\tau L = 0 \Longrightarrow \tau = \frac{1 - y_t w x_t}{\left|\left|x_t\right|\right|^2}$$

זה קונסיסטנטי עם האלגוריתם הדואלי.

<u>טענה</u> - אחרי עדכון, w_{t+1} יוכל לתייג נכון את x_t, כלומר $w_{t+1}x_t y_t > 0$.
<u>הוכחה</u> -

$$y_t w_{t+1} x_t = y_t(w_t + \tau x_t y_t)x_t = y_t w_t x_t + \tau\left|\left|x_t\right|\right|^2 y_t^2 = y_t w_t x_t + \frac{1 - y_t w x_t}{\left|\left|x_t\right|\right|^2}\left|\left|x_t\right|\right|^2 y_t^2 =$$
$$= y_t w_t x_t + (1 - y_t w x_t) \cdot 1 = 1 > 0$$

כלומר, הוא לא רק תייג נכון אלא גם מצא margin שהוא בדיוק 1. ■

<u>הפיכת אלגוריתם Online ל- Batch</u>

כדי להפוך אלגוריתם online לאלגוריתם batch (להתייחס למה שהיה עד עכשיו כאימון), לא ניתן להשתמש ב- $w^* = w^m$, כי הוא עלול להיות מוטה (היפותזה נעה – משתנה עם הזמן). פתרונות:

- <u>Random Shuffle + ממוצע (Averaging)</u> – מסכמים גם עבור w^t שעבורם לא היה עדכון.

$$w^* = \frac{1}{T}\sum_{t=0}^{T-1} w^t$$

- בכל שלב, בודקים את w_t על validation set ושומרים את ה- w_t עם השגיאה המינימלית.

סיווג למחלקות מרובות (Multiclass)

הגדרת הבעיה

נתונות m דוגמאות: $S = \{(x_i, y_i)\}$, כאשר $x \in R^d$ ו- $y \in \{1, \dots, k\}$. אנחנו מדברים על $k > 2$, כלומר הסיווג הוא לא בינארי, אלא כל דוגמה מסווגת לאחת מכמה מחלקות. אם $k = \infty$ אז לא נתון מראש כמה מחלקות יש וכל דוגמה חדשה יכולה להיות שייכת למחלקה שלא ראינו קודם.

המטרה היא למזער את הטעות: $\min\big(P(\hat{y} \neq y)\big) = \min|E[\delta(\hat{y} \neq y)]|$.

פתרונות לבעיית ה-Multiclass

רדוקציה לבעיה הבינארית

- One vs. All (OvA)
בשלב האימון, מחלקים את הבעיות לבעיה של 1 מול $k, \dots, 2$, בעיה של 2 מול $k, \dots, 1,3,4$ וכו', עד k מול $k-1, \dots, 1$. כל אחת מהבעיות היא SVM בינארי, כאשר המחלקה הראשונה היא +1 ושאר המחלקות הן 1-. בסה"כ, פותרים k בעיות SVM בלתי תלויות. בשלב ההסקה, בוחרים $y = argmax_{r=i\dots k}(w^r x)$, כאשר w^r ווקטור לכל מחלקה. קיימות מס' בעיות בשיטה זו:
הבעיה הראשונה היא בעיה חישובית - צריך לפתור k בעיות. בעיה נוספת היא שכאשר מס' הדוגמאות אינו מאוזן לכל מחלקה, SVM לא עובד טוב (הוא מוטה לטובת המחלקה שיש לה יותר דוגמאות), וזה יקרה כמעט תמיד ב- OvA (רוב הדוגמאות יהיו בצד של all).
בבעיה בינארית, $y \in \{-1, +1\}$, אז יהיה w^{-1} ו- w^{+1} כאשר $w^{-1} = -w^{+1}$.
$$y = argmax_{r\in\{-1,+1\}}(w^r x) = \begin{cases} -1 & w^{-1}x > w^{+1}x \\ +1 & else \end{cases} = sign(w^{+1}x)$$
העדכון של Perceptron (One vs. All) שומר על הסימטריה בין הווקטורים: אם ראינו דוגמה x חיובית, אז צריך לעדכן את הווקטורים בהתאם, ונקבל:
$w^{+1} = w^{+1} + yx = w^{+1} + x$ ו- $w^{-1} = w^{-1} - yx = w^{-1} - x = -w^{+1}$.

- All-pairs (AP)
בשלב האימון, משווים את המחלקות 1-2, 1-3, וכו' עד $k, k-1$. יש $\binom{k}{2} = \frac{k(k-1)}{2}$ בעיות SVM בינאריות בלתי תלויות לפתור. היתרון של השיטה הזו לעומת הקודמת הוא שמס' הדוגמאות מאוזן. גם השיטה הזו בעייתית מבחינה חישובית - צריך לפתור $O(k^2)$ בעיות. בשלב ההסקה, אפשר להשתמש ב- $y = argmax_{r=i\dots k} sign(w^r x)$. אם כל דוגמה מסווגת בדיוק למחלקה אחת, זה יעבוד טוב. דרך אחרת היא למנות כמה פעמים כל מחלקה הייתה חיובית ולבחור את זו שהייתה הכי הרבה פעמים חיובית: $y = argmax_{r=i\dots k}|\{r: sign(w^r x) = +1\}|$.
השיטה הכי טובה היא: $y = argmax_{r=i\dots k} \sum_r w^r x$ – לסכום את הציונים של כל מחלקה ולבחור את המחלקה עם הציון הכי גבוה (באופן דומה ל- OvA). השיטה עובדת טוב.

- Error Correcting Output Codes (ECOC)
שיטה שמאחדת את שיטת השיטות הקודמות.
מגדירים מטריצה לבעיות הבינאריות: $M \in \{-1,0,+1\}^{k\times l}$ כאשר l – מס' המסווגים ו- k – מס' המחלקות. לדוגמה, ב- OvA, תהיה לנו מטריצה $k \times k$ כאשר באלכסון יהיה +1 ובכל שאר המטריצה יהיה 1-. ב- AP, תהיה לנו מטריצה $k \times \frac{k(k-1)}{2}$ כאשר בכל מסווג (טור) תהיה בדיוק מחלקה אחת (שורה) עם +1, מחלקה אחת (שורה) עם 1- ושאר המחלקות (שורות) עם 0.
באופן כללי, $M(r,s)$ הוא הערך של המחלקה ה- r במסווג ה- s.

מריצים כל מסווג בינארי s פעם אחת על כל דוגמה x ומרכיבים את המטריצה $f(x)$ בה $f_s(x)$ היא התוצאה מהמסווג ה- s: $w^s x$. בשלב ההסקה, משתמשים במרחק Hamming (מרחק בין שתי סדרות בינאריות – 0 אם המספרים זהים ו- 1 אחרת), ובוחרים:

$$\hat{y} = argmin_r d_H\big(M(r), f(x)\big) = argmin_r \sum_{s=1}^{l} \left(\frac{1 - sign[M(r,s) f_s(x)]}{2} \right)$$

לכל שורה (מחלקה), הסכום הוא על כל המסווגים (טורי המטריצה) ואנחנו סוכמים את הטעות, ובוחרים את המחלקה עם הטעות הכי קטנה, כלומר שהשורה שלה ב-M הכי קרובה לשורה ב-f.
אם $M(r,s), f_s(x)$ שווי סימן, זה לא תורם כלום לסכום הטעויות. אם הם שוני סימן, זה יתרום 1 לסכום, ואם אחד מהם הוא 0, זה יתרום $\frac{1}{2}$ לסכום.

אפשר גם להסתכל על "החלטה רכה" שלוקחת בחשבון את הגודל ולא רק את הסימן:

$$\hat{y} = argmin_r \sum_{s=1}^{l} l[M(r,s) f_s(x)] = argmin_r \sum_{s=1}^{l} \max\{1 - M(r,s) w^s x, 0\}$$

(דוגמה מפורטת במאמר על Multiclass).

<u>Multiclass Perceptron</u>

מוצאים סט ווקטורי המשקלים, ווקטור לכל מחלקה: $w = \{w^1, \dots w^k\}$.
השגיאה לא משתנה. כלל העדכון במקרה של שגיאה ($\hat{y} \neq y_i$) הוא:

$$w^{y_i} = w^{y_i} + x_i$$
$$w^{y'} = w^{y'} - x_i$$
$$w^{y} = w^{y}, y \neq y_i, y'$$

כלומר, מגדילים את הווקטור של המחלקה האמיתית, מקטינים את הווקטור של המחלקה שחזינו בטעות, ולא משנים את הווקטורים של שאר המחלקות.

אפשר אחרת היא להגדיר את סט המחלקות שהציון שהן נותנות לדוגמה x גבוה יותר מהציון שנותנת המחלקה האמיתית y:

$$E_y = \{r \neq y | w^r x > w^y x\}$$

ולעדכן את ווקטורי המשקלות כך שמקטינים את הווקטורים של כל המחלקות שנתנו ציון גבוה יותר לדוגמה, ולא רק את החיזוי עצמו, באופן הבא:

$$w^{y_i} = w^{y_i} + x_i$$
$$w^{y} = w^{y} - \frac{x_i}{|E_i|}, y \in E_i$$
$$w^{y} = w^{y}, y \neq y_i, y \notin E_i$$

Multiclass SVM

מוצאים סט ווקטורי המשקלים, ווקטור לכל מחלקה: $w = \{w^1, \dots w^k\}$.

השגיאה לא משתנה, ומוגדרת להיות $Err \equiv P[\hat{y} \neq y] = E[\delta(\hat{y} \neq y)]$.

המטרה היא עדיין להביא למינימום את תוחלת השגיאות: $w^* = argmin_{\{w^*\}} E_{(x,y)}[\delta(\hat{y} \neq y)]$.

מכיוון שאין לנו את ההתפלגות האמיתית של דוגמאות האימון, לוקחים ממוצע על סט האימון ומוסיפים גורם רגולריזציה:

$$w^* = argmin_{\{w^*\}} \frac{1}{m} \sum_{i=1}^{m} \delta(\hat{y} \neq y_i) + \frac{\lambda}{2} \sum_{r=1}^{k} ||w_r||^2$$

החיזוי מתבצע ע"י: $\hat{y} = arg\max_{\hat{y}} w^{\hat{y}} x$.

גם במקרה הזה, משתמשים באלגוריתם SSGD הפותר באופן כללי בעיית $w^* = argmin_w(F(w))$, כאשר $F(w) = \frac{1}{m} \sum_{i=1}^{m} f_i(w)$, בו מוסיפים בכל איטרציה ל- w את התת-גרדיאנט של הדוגמה הנוכחית.

כפי שראינו, יש למצוא חסם עליון קמור ל- 0-1 Loss, שאינה קמורה:

$$\delta(\hat{y} \neq y_i) = \delta(\hat{y} \neq y_i) - w^y x + w^y x \leq \delta(\hat{y} \neq y_i) - w^y x + \max_{\hat{y}} w^{\hat{y}} x =$$

לפי הבחירה של $\hat{y}$ (כ- argmax):

$$= \delta(\hat{y} \neq y_i) - w^y x + w^{\hat{y}} x \leq \max_{y'} [\delta(y' \neq y_i) - w^y x + w^{y'} x]$$

לכן ניתן להציב במקום:

$$w^* = argmin_{\{w^*\}} \frac{1}{m} \sum_{i=1}^{m} \max_{y'} \left[\delta(y' \neq y_i) - w^{y_i} x_i + w^{y'} x_i\right] + \frac{\lambda}{2} \sum_{r=1}^{k} ||w^r||^2$$

פונקציית ה-Loss החדשה היא: $\max_{y'} \left[\delta(y' \neq y_i) - w^{y_i} x_i + w^{y'} x_i\right]$, הנקראת Surrogate Loss.

במקום $\delta(y' \neq y_i)$ אפשר להגדיר פונקציית Loss שהיא user defined שנותנת משקלים שונים לטעויות שונות. למשל, בזיהוי דיבור, שגיאה בין aa ל-ae ושגיאה בין aa ל-d הן שגיאות שונות (הראשונה פחות חמורה מהשנייה).

נתבונן בתת-גרדיאנטים של w^*:

$$[\nabla_{w^{y_i}}]_i = -x_i + \frac{\lambda}{2} 2||w^{y_i}|| = -x_i + \lambda w^{y_i}$$

$$\left[\nabla_{w^{y'}}\right]_i = +x_i + \frac{\lambda}{2} 2 \left|\left|w^{y'}\right|\right| = +x_i + \lambda w^{y'}$$

$$[\nabla_{w^r}]_i = \frac{\lambda}{2} 2||w^r|| = \lambda w^r, r \neq y_i, y'$$

האלגוריתם (SSGD):

```
for t=1…T
    choose (x_i, y_i) ∈ S
        y' = argmax_{y'∈{1,…,k}}[δ(y' ≠ y_i) + w^{y'} x_i]
        if [δ(y' ≠ y_i) − w^{y_i} x_i + w^{y'} x_i] > 0:
            w^{y_i} = w^{y_i} − η_t ∇_{w^{y_i}} = w^{y_i}(1 − η_t λ) + x_i η_t
            w^{y'} = w^{y'} − η_t ∇_{w^{y'}} = w^{y'}(1 − η_t λ) − x_i η_t
            w^r = w^r − η_t ∇_{w^r} = (1 − η_t λ) w^r   for , y ≠ y_i, y'
```

לוקחים דוגמה (x_i, y_i) ומסווגים אותה למחלקה y'. מפחיתים מכל ווקטור מחלקה את התת-גרדיאנט של פונקציית המטרה לפי הווקטור הזה (כדי לשנות את הגודל שלו מבלי לשנות כיוון). מקבלים שלכל y שאינה המחלקה האמיתית (y_i) או הפרדיקציה (y'), מקטינים קצת את הווקטור. את הווקטור של המחלקה האמיתית מגדילים קצת ואת הווקטור של מחלקת הפרדיקציה מקטינים עוד קצת. η הוא קבוע הלמידה הקובע כמה משקל לתת לדוגמה הנוכחית ו- λ הוא קבוע הרגולריזציה.

CRF (Conditional Random Fields)
הכללה של Logistic Regression ל-Multiclass. הגדרת ההסתברות היא:

$$P_w(Y = y|X = x) = \frac{e^{w^y x}}{\sum_{y'=1}^{k} e^{w^{y'} x}}$$

כאשר w^y ווקטור המשקלות של מחלקה y.

רוצים למצוא w בשיטת MLE ולכן ממזערים את מינוס ה-log-likelihood.
בעיית האופטימיזציה על סט האימון, עם גורם רגולריזציה, מוגדרת כך:

$$w^* = argmin_w \frac{1}{m}\sum_{i=1}^{m} -\ln\left(P_w(y_i|x_i)\right) + \frac{\lambda}{2}\sum_{y=1}^{k}\left||w^y|\right|^2 =$$

$$= argmin_w \sum_{i=1}^{m} -\ln\left(\frac{e^{w^y x}}{\sum_{y'=1}^{k} e^{w^{y'} x}}\right) + \frac{\lambda}{2}\sum_{y=1}^{k}\left||w^y|\right|^2 =$$

$$= argmin_w \sum_{i=1}^{m} -\left(\ln\left(e^{w^y x}\right) - \ln\left(\sum_{y'=1}^{k} e^{w^{y'} x}\right)\right) + \frac{\lambda}{2}\sum_{y=1}^{k}\left||w^y|\right|^2 =$$

$$= argmin_w \sum_{i=1}^{m} -\left(w^{y_i} x_i - \ln\left(\sum_{y=1}^{k} e^{w^y x_i}\right)\right) + \frac{\lambda}{2}\sum_{y=1}^{k}\left||w^y|\right|^2 =$$

$$= argmin_w \sum_{i=1}^{m} \left(\ln\left(\sum_{y=1}^{k} e^{w^y x_i}\right) - w^{y_i} x_i\right) + \frac{\lambda}{2}\sum_{y=1}^{k}\left||w^y|\right|^2$$

העדכון לווקטורי המשקלות באלגוריתם SSGD מוגדר כך:

$$w^y = w^y - \eta \nabla_{w^y}\left[-\ln\left(P_w(y_i|x_i)\right) + \frac{\lambda}{2}\sum_{y=1}^{k}\left||w^y|\right|^2\right]$$

כאשר η הוא קבוע הלמידה. נחשב את הגרדיאנט:

עבור $r \neq y_i$:

$$\nabla_{w^r}\left[-\ln\left(P_w(y_i|x_i)\right)+\frac{\lambda}{2}\sum_{y=1}^{k}\left||w^y|\right|^2\right]=\nabla_{w^r}\left[-w^{y_i}x_i+\ln\left(\sum_{y=1}^{k}e^{w^yx_i}\right)+\frac{\lambda}{2}\sum_{y=1}^{k}\left||w^y|\right|^2\right]=$$
$$\frac{x_i\cdot e^{w^rx_i}}{\sum_{y'=1}^{k}e^{w^{y'}x_i}}+\lambda w^r=x_i\cdot P_w(r|x_i)+\lambda w^r$$

עבור y_i:

$$\nabla_{w^{y_i}}\left[-\ln\left(P_w(y_i|x_i)\right)+\frac{\lambda}{2}\sum_{y=1}^{k}\left||w^y|\right|^2\right]=\nabla_{w^{y_i}}\left[-w^{y_i}x_i+\ln\left(\sum_{y=1}^{k}e^{w^yx_i}\right)+\frac{\lambda}{2}\sum_{y=1}^{k}\left||w^y|\right|^2\right]$$
$$=-x_i+\frac{x_i\cdot e^{w^{y_i}x_i}}{\sum_{y'=1}^{k}e^{w^{y'}x_i}}+\lambda w^{y_i}=-x_i+x_i\cdot P_w(y_i|x_i)+\lambda w^{y_i}=x_i(P_w(y_i|x_i)-1)+\lambda w^{y_i}$$

נציב בכלל העדכון ונקבל:

$$w^r=w^r-\eta(x_i\cdot P_w(r|x_i)+\lambda w^r)=w^r(1-\eta\lambda)-\eta x_iP_w(r|x_i)$$
$$w^{y_i}=w^{y_i}-\eta(x_i(P_w(y_i|x_i)-1)+\lambda w^{y_i})=w^{y_i}(1-\eta\lambda)-\eta x_i(P_w(y_i|x_i)-1)$$

ההסקה ב- Multi-class Logistic Regression מוגדרת ע"י:

$$\hat{y}_l=argmax_{y\in\{1,\dots,k\}}w^yx$$

נשים לב שההסקה תהיה זהה לזו של מסווג בייסיאני. במסווג בייסיאני, הסיווג נעשה ע"י החוק:

$$\hat{y}_b=argmax_{y\in\{1,\dots,k\}}P(Y=y|x)$$

מהמונוטוניות של פונקציית ln:

$$\hat{y}_b=argmax_{y\in\{1,\dots,k\}}P(Y=y|x)=argmax_y\ln\left(P(Y=y|x)\right)=$$

נציב את ההתפלגות:

$$=argmax_y\left(w^yx-\ln\left(\sum_{y'=1}^{k}e^{w^{y'}x}\right)\right)=$$

המחובר $\ln\left(\sum_{y'=1}^{k}e^{w^{y'}x}\right)$ הוא קבוע ביחס ל- y, לכן:

$$=argmax_yw^yx=\hat{y}_l$$

דוגמאות קוד

דוגמאות הקוד עוסקות בסיווג OCR של הספרות 1-8. קובץ האימון הוא בפורמט שבכל שורה יש דוגמה אחת. המספר הראשון הוא ה- Y label (1-8) ושאר המספרים הם X – ערכי הפיקסלים כווקטור.

```
import numpy as np
import matplotlib.pyplot as plt
import random

# read data
training_data = np.loadtxt("training_data_1_vs_8.rs.dat.gz");
X = training_data[:,1:]
Y = training_data[:,0]
```

בשיטת All pairs, כשרוצים להשוות בין 1 ל-8, הופכים את ה- label של 8 ל-(-1):

```
# convert 1 to +1 and 8 to -1
Y_binary = np.where(Y == 1, 1, -1)
m, d = X.shape;
```

הקוד מדפיס 25 דוגמאות, את הדוגמאות החיוביות בצבע המקורי ואת הדוגמאות השליליות בנגטיב.

```
# show some images
plt.figure(1);
for i in range (1,26):
    ax = plt.subplot(5,5,i);
    ax.axis('off');
    if Y_binary[i] > 0:
        ax.imshow(X[i,:].reshape(28,28), cmap="gray");
    else:
        ax.imshow(255-X[i,:].reshape(28,28), cmap="gray");
plt.draw();
```

הקוד של פרספטרון מאתחל את w=0 ומס' השגיאות M=0. הוא רץ 2000 איטרציות, בוחר דוגמה באופן אקראי, מנבא את ה- label שלה ע"י הסימן של wx. אם זו טעות, מעדכנים את w ומגדילים את מס' השגיאות.

```
# weight vector
w = np.zeros((d,));
M = 0; # counts mistakes

# Perceptron
T = 2000
for t in range(0, T):
    # choose example
    i = random.randint(0, m-1)
    # predict
    Yhat = np.sign(np.dot(w, X[i,:]))
    if Y_binary[i] != Yhat:
        M = M + 1
        # update
        w = w + Y_binary[i]* X[i,:]
w_perceptron = w
```

באופן עקרוני, בפרספטרון צריך לעבור על כל הדוגמאות. הרבה פעמים מבצעים כמה מעברים על הדוגמאות, כאשר כל מעבר נקרא epoch (T epochs). בדוגמה שראינו יש ערבוב של שני הדברים.

הסיבה לכך היא שפרספטרון באופן עקרוני הוא אלגוריתם אונליין. ההיפותזה יכולה להשתנות (למשל, בסינון ספאם, יכול להיות שתחומי העניין שלנו השתנו ומה שהיה פעם ספאם הוא כבר לא או להיפך). ב- batch יש סדרת אימון וסדרת בדיקות, וב- online יש רק סדרת אימון. כדי להפוך אלגוריתם אונליין לאלגוריתם batch, עושים random shuffle מראש לדוגמאות או שבוחרים באופן רנדומלי בכל איטרציה.

הקוד הבא מדפיס את w:

```
# show the mask learnt by Perceptron
plt.figure(2);
ax1 = plt.subplot(1,2,1);
ax1.axis('off'); # no need for axis marks
ax2 = plt.subplot(1,2,2);
ax2.axis('off'); # no need for axis marks
ax1.imshow(w.reshape(28,28),cmap="gray");
tmp = 1/(1+np.exp(-10*w/w.max()));
ax2.imshow(tmp.reshape(28,28),cmap="gray");
plt.draw();
```

w הוא בגודל של x, כלומר בגודל התמונה. אפשר לצפות בו בתור תמונה. אפשר לזהות בו (בקושי) את הספרה 1 בצבע לבן (חיובי) ואת הספרה 8 בצבע שחור (שלילי). שאר התאים הם אפורים (0). הסיבה לכך שפיקסלים שיוצרים את הספרה 1 אמורים לתרום מס' חיובי לתחום ופיקסלים שיוצרים את הספרה 8 אמורים לתרום מס' שלילי לסכום. כל הפיקסלים הצדדיים לא משפיעים על הסיווג, לכן ערכם הוא 0 ו-w הוא ווקטור דליל.

הקוד של SVM מאתחל את w=0 ומס' השגיאות M=0. הוא רץ 2000 איטרציות, בוחר דוגמה באופן אקראי, מנבא את ה- label שלה ע"י הסימן של wx. מגדירים את ה- loss ל- $\max\{0,1-y_i w x_i\}$. זה חסם עליון קמור לשגיאה $1\{y_i w x_i < 0\}$. אם זו טעות (ה-loss גדול מ-0), מעדכנים את w לפי האלגוריתם של Hildreth (עם τ, α) ומגדילים את מס' השגיאות. אפשר גם לשנות את הקוד שישתמש ב- SGD.

```
# weight vector
w = np.zeros((d,));
M = 0; # counts mistakes

# Support vector machines
T = 2000
alpha = np.zeros((m,))
for t in range(0, T):
    # choose example
    i = random.randint(0,m-1)
    # predict
    Yhat = np.sign(np.dot(w, X[i,:]))
    # compute hinge loss
    loss = max([0.0, 1.0- Y_binary[i]*np.dot(w, X[i,:])])
    if loss > 0.0:
        M = M + 1
        # update w
        tau = max([-alpha[i], loss/np.dot(X[i,:], X[i,:])])
        alpha[i] = alpha[i] + tau
        w = w + tau*Y_binary[i]* X[i,:]
w_svm = w
```

צפינו גם ב- w שהתקבל מה- SVM והוא היה די דומה, אבל הפרספטרון היה קצת יותר רועש. ב- SVM יש יותר שגיאות, כי בפרספטרון סופרים שגיאה רק כשהמסווג לא נכון בזמן האימון: $\hat{y} \neq y_i$, וב- SVM דורשים $y_i w x_i < 1$, כלומר לא רק שהוא יהיה בצד הנכון אלא גם עם margin, וזו דרישה יותר חריפה ולכן יש יותר שגיאות.

ראינו את אחוזי השגיאה על test set וראינו שהאחוזים היו מאוד נמוכים (1.5-2) ובדר"כ לטובת ה- SVM.

```
# check performence on test data
test_data = np.loadtxt("test_data_1_vs_8.dat.gz");
X = test_data[:,1:]
Y = test_data[:,0]

# convert 3 to +1 and 8 to -1
Y_binary = np.where(Y == 1, 1, -1)
m, d = X.shape;

M_perceptron = 0
for t in range(0, m):
    Yhat = np.sign(np.dot(w_perceptron, X[t,:]))
    if Y_binary[t] != Yhat:
        M_perceptron = M_perceptron + 1
print "perceptron err=", float(M_perceptron)/m

M_svm = 0
for t in range(0, m):
    Yhat = np.sign(np.dot(w_svm, X[t,:]))
    if Y_binary[t] != Yhat:
        M_svm = M_svm + 1
print "SVM err=", float(M_svm)/m
```

כדי לראות את התוצאות בצורה גרפית, יוצרים מידע סינטטי. יוצרים מטריצה X עם 1 ו- 0 באופן רנדומלי והופכים אותם ל- 1 ו- (-1). מסווגים לפי ה- w שהתקבל ועוברים על הדוגמאות האלה. לכל דוגמה, אם ה- margin יצא פחות מחצי, מוחקים אותה.

```
# generate data
m = 1000
d = 2
desired_num_pos = 10
w_opt = np.array([1,2])
# random numbers in [0,1]
X = np.random.rand(m,d)
# convert to numbers in [-1,1]
X = 2.0*X - 1.0
# set the label
Y  = np.zeros((m,))
num_pos = 0
indices_to_delete = list()
for i in range(m):
    Y[i] = np.sign(np.dot(w_opt, X[i,:]))
#     # set margin
#     if Y[i]*np.dot(w_opt, X[i,:]) < 0.5:
#         indices_to_delete.append(i)
#
#X = np.delete(X, indices_to_delete, 0)
#Y = np.delete(Y, indices_to_delete, 0)
m, d = X.shape;  # updated number of examples
```

מציירים את הגרף עם הדוגמאות החיוביות בירוק והשליליות באדום. אפשר לראות את ההפרדה. הרצנו את האלגוריתם של SVM וציירנו את הדוגמאות עם המפריד. הרצנו גם את פרספטרון וראינו ש- w יצא שונה ולא התחשב ב- margin (כי פרספטרון לא מתייחס ל- margin). הרצנו כמה פעמים. הפרספטרון השתנה ו- SVM לא. הסיבה לכך היא ש- SVM פותר בעיית אופטימיזציה. אם יש לנו מספיק דוגמאות, הוא יתכנס תמיד לאותו המפריד. זה לא קשור ל- margin, אלא לעדכון של w. גם כשהוספנו margin לפרספטרון, עדיין הוא השתנה מריצה לריצה.

דיברנו על כך שה- margin=1 ב- SVM הוא סימבולי ושאין משמעות ל-1. כששינינו את ה- margin, ראינו שקיבלנו את אותה התוצאה. נניח שלקחנו margin מאוד גדול. על כל טעות אנחנו מעדכנים את w. אם נעדכן אותו מספיק פעמים, בסוף w ייתן את ה- margin הרצוי. w ייצא עם ערכים גדולים יותר אבל היחס בין הקואורדינטות השונות שלו נשמר.

```
# add noise
for i in range(m):
    if i % 10 == 0:
        Y[i] = -Y[i]
```

רצינו לראות מה ההשפעה של הפרמטר C. כזכור, C הוא פרמטר ב- SVM שקובע כמה למזער את ה- margin וכמה להתייחס לטעויות. אם C קטן, יותר חשוב ה- margin מאשר למזער את הטעויות, ואם C גדול, חשוב למזער את הטעויות יותר מאשר את ה- margin. כדי לראות את ההשפעה של C, הוספנו רעש למידע. ראינו שבצד החיובי יש קצת דוגמאות שליליות ולהפך. שהגדלנו את C, הדוגמאות הרועשות פחות השפיעו על הפתרון, ולהיפך. כשפותרים SVM, הפתרון צריך להיות בתוך כל האילוצים (בצורה גרפית). בפועל, הוא תמיד יהיה על אילוץ והאילוץ הזה יהיה הווקטור התומך. המינימום גורם לכך שבדר"כ הפתרון יהיה בחיתוך של שני אילוצים. האלגוריתם האיטרטיבי מטיל על האילוצים. הפרמטר C מגביל אותו מלהתקרב לאילוצים. הוא גורם לו לפתור פחות מהאילוץ, לדוגמה: $1-\xi \le ywx-1$ במקום $ywx \le 1-1$. אם יש דוגמאות רועשות, ה- SVM יתקרב אליהן פחות.

Multilabel

הגדרת הבעיה

זוהי הרחבה של Multiclass, בה ניתן לסווג כל דוגמה לכמה מחלקות. לדוגמה x יש סט מחלקות Y. רוצים למצוא פונקציה שבהינתן x תחזיר את Y. קשה למצוא פונקציה כזו, מכיוון ש- Y הוא בגודל משתנה. הפתרון הוא להחזיר את כל המחלקות ממוינות בסדר יורד. נסמן $y^* = y \cdot \ldots \cdot y$. מתקיים:

$$Y = f(x) = \arg sort_{r \in y^*} w^r x$$

כלומר, Y ווקטור של מחלקות $r \in y^*$ ממוין לפי $w^r x$. בשיטה זו אין סף מינימום וכל המחלקות יופיעו בווקטור אך בסדר יורד ועם ציון.

כיצד נגדיר את פונקציית ה-Loss? נתון הווקטור הממוין (ה-"label" שחזינו), ותיוג אנושי הקובע לכל מחלקה בווקטור האם היא רלוונטית לדוגמה או לא (ה-"label" האמיתי). היינו רוצים שבראש הווקטור יופיעו כמה שיותר מחלקות רלוונטיות ושהמחלקות הרלוונטיות יופיעו בעיקר בראש הווקטור. מגדירים את ה- margin בתור הפרש בין המחלקה הלא רלוונטית הכי גבוהה בדירוג לבין המחלקה הרלוונטית הכי נמוכה בדירוג:

$$margin = \min_{r \in Y_i} w^r x_i - \max_{s \notin Y_i} w^s x_i$$

בדירוג האופטימלי, ה- margin יצא 1, כי כל הדוגמאות החיוביות מופיעות למעלה ומיד אחריהן כל השליליות. בכל דירוג אחר, השוליים יצאו שליליים, והיינו רוצים שהם יהיו כמה שיותר גדולים (כמה שפחות שליליים). ה- loss מוגדר להיות $l(w, x_i, Y_i) = \left[1 - \min_{r \in Y_i} w^r x_i + \max_{s \notin Y_i} w^s x_i\right]_+$

כאשר $[.]_+ \coloneqq \max\{0, .\}$. בדירוג האופטימלי נקבל $l(w, x_i, Y_i) = 0$. בכל דירוג אחר נקבל margin שלילי ולכן $l(w, x_i, Y_i) > 1$.

<u>דוגמה מנחה:</u> חלוקה של מסמכים לנושאים. יתכן שמסמך עוסק ביותר מנושא אחד.

x_i הוא ווקטור המאפיינים של המסמך ה-i, והוא מכיל מאפיין עבור כל מילה במסמך. הערך של המאפיין אמור לשקף את חשיבות המילה במסמך. אם מספר המסמכים סופי, ניתן להשתמש במדד TF-IDF, שנותן למילה ציון עפ"י השכיחות שלה במסמך לעומת השכיחות בשאר המסמכים. הרעיון של מדד זה הוא שאם המילה מופיעה בשכיחות גבוהה במסמך, אך אינה מופיעה בשכיחות גבוהה בשאר המסמכים, החשיבות שלה גדולה עבור המסמך, ולכן היא תקבל ציון גבוה. לעומת זאת, מילים נפוצות (כמו "את", "גם") יקבלו ציון נמוך כי הן מופיעות בתדירות גבוהה בכל המסמכים. פירוט לגבי המדד נמצא בפרק על מנועי חיפוש. ב- web כאשר מספר המסמכים שלנו אינסופי, לא משתמשים ב- TF-IDF, כי אי אפשר למדוד תדירות של מילים בכל ה- web, לכן שולפים מראש רק את המסמכים שמכילים את מילות החיפוש.

פתרונות לבעיית ה-Multilabel

SVM ל- Multilabel

נשנה את הנוסחה הרגילה: $w^* = argmin_w \frac{1}{m}\sum_{i=1}^{m} l(w, x_i, y_i) + \frac{\lambda}{2}\left|\left|w\right|\right|^2$, לנוסחה:

$$\{w^k\}^* = argmin_{\{w^k\}} \frac{1}{m}\sum_{i=1}^{m} l\big(\{w^k\}, x_i, Y_i\big) + \frac{\lambda}{2}\sum_{i=1}^{k}\left|\left|w^k\right|\right|^2$$

כלומר, עדיין מנסים למזער את ה-loss על החיזוי, והרגולריזציה היא על כל ווקטורי המשקלים w.

האלגוריתם:

```
Loop
    r = argmin_{r∈Y_i} w^r x_i   מוצא את הרלוונטי הכי נמוך
    s = argmax_{s∉Y_i} w^s x_i   מוצא את הלא רלוונטי הכי גבוה
    update:
    w^r ← w^r(1 − ηλ) + x_i η
    w^s ← w^s(1 − ηλ) − x_i η
    w^k ← w^k(1 − ηλ) ∀k ≠ r, s
```

כלומר, בכל שלב הוא מוצא את המחלקה הרלוונטית המדורגת הכי נמוך ומגדיל את ווקטור המשקלות שלה, מוצא את המחלקה הלא רלוונטית המדורגת הכי גבוה ומקטין את ווקטור המשקלות שלה, ומקטין קצת גם את שאר ווקטורי המשקלות. ישנה גם גרסה קונסרבטיבית בה לא מעדכנים את ווקטורי המשקלות האחרים.

בפרספטרון, זה יהיה דומה עם $\eta = 1$ וללא רגולריזציה.

אם רוצים לדעת רק אילו מחלקות רלוונטיות מתוך $\hat{y}$, משנים: $\hat{y} = \arg sort_{r\in y^*}[w^r x - S(x)]_+$, כאשר $S(x)$ הן המחלקות הרלוונטיות, ומחזירים נחזיר רק מחלקות שהציון שלהם יצא חיובי. לצורך כך, מגדירים מחדש: $l(w, x_i, y_i) = \max_{r\in Y_i}[1 + S(x) - w^r x] + \max_{s\notin Y_i}[1 - S(x) + w^s x]$.

דוגמאות קוד

הקוד הבא הוא דוגמה לסיווג למחלקות מרובות. בדוגמה מסווגים תמונות לפי OCR לספרות 1,8,4. בשלב ראשון, קוראים את המידע מהקובץ. כל שורה היא דוגמת אימון. התו הראשון הוא ה-label ושאר השורה היא המאפיינים. מס' המחלקות נקבע לפי ה-label הגבוה ביותר בדוגמאות האימון.

```python
import numpy as np
import matplotlib.pyplot as plt
import random

# read data
data = np.loadtxt("training_data_1_vs_8_vs_4.rs.dat.gz");
X = data[:,1:]
Y = data[:,0]
m, d = X.shape;  # m number of examples, d instance dimesion
k = max(Y)+1  # k number of classes
```

ה-SVM רץ 6000 איטרציות. בכל איטרציה הוא בוחר דוגמה אקראית ומנבא את המחלקה לפי המחלקה r שנותנת את הציון המקסימלי $w^r x$. המטריצה w מכילה בשורה ה- i את וקטור המשקלות w^i. פונקציית ה- loss מוגדרת להיות $\left[1-w^{y_i}x+w^{\hat{y}}x\right]$.$loss(w,x_i,y_i)=$ רוצים שיתקיים $w^{y_i}x \geq w^{\hat{y}}x$ אבל מוסיפים 1 (margin) כדי להגן מפני רעש. אם ה- $loss > 0$, כלומר יש שגיאה בחיזוי, מעדכנים את מס' השגיאות וב- w מגדילים את המחלקה האמיתית ומקטינים את יתר המחלקות (בדוגמה זו יש שימוש ב- τ שנובע מהגדרת הבעיה הדואלית).

```python
# weight vector
w = np.zeros((k, d));
M = 0; # counts mistakes

# Support vector machines
T = 6000
alpha = np.zeros((m,))
for t in range(0, T):
    # choose example
    i = random.randint(0,m-1)
    # predict
    Yhat = np.argmax(np.dot(w, X[1,:]))
    # compute hinge loss
    loss1 = max([0.0, 1.0 - np.dot(w[Y[i],:], X[i,:]) +
np.dot(w[Yhat,:], X[i,:])])
    if loss1 > 0.0:
        M = M + 1
        # update w
        tau = loss1/np.dot(X[i,:], X[i,:])
        w[Y[i],:] = w[Y[i],:] + tau*X[i,:]
        w[Yhat,:] = w[Yhat,:] - tau*X[i,:]
```

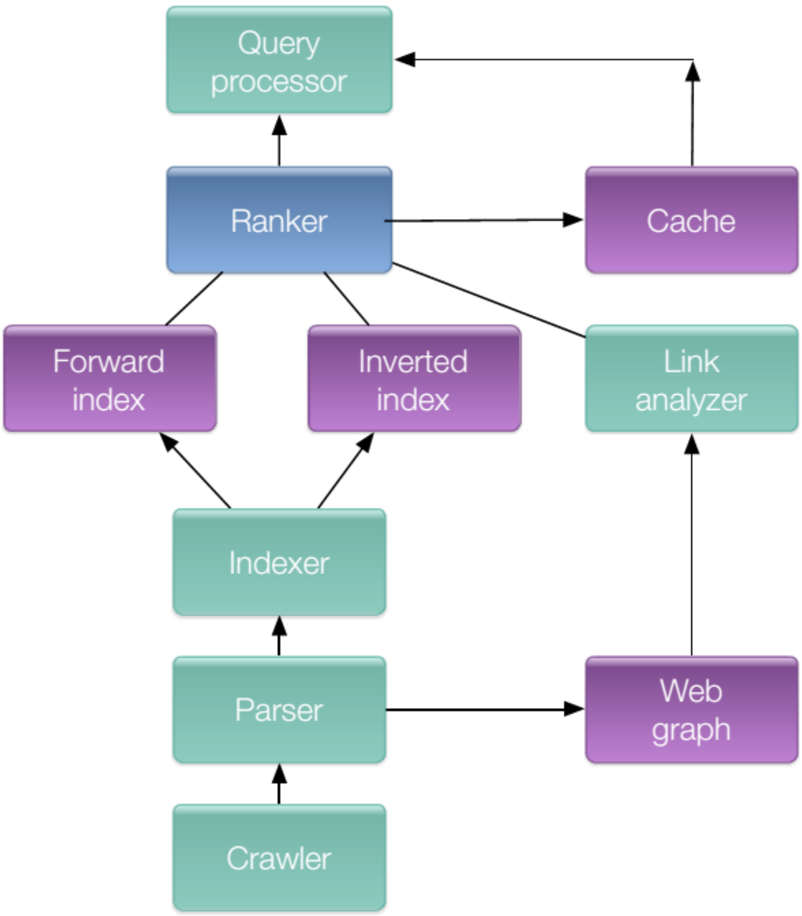
ראינו את ה- w^i השונים בצורה ויזואלית כאשר כל אחד מהם הכיל את הספרה שלו בצבע לבן ואת שתי הספרות האחרות בצבע שחור.

```python
plt.figure(2);
classes = [1, 4, 8]
for i, c in enumerate(classes):
    print i, c
    ax1 = plt.subplot(1,3,i+1);
    ax1.axis('off');
    ax1.imshow(w[c,:].reshape(28,28),cmap="gray");
plt.draw();
```

ה- 1 בפונקציית ה- loss יכול להיות מוחלף ב- $\gamma(y,\hat{y})$ שנותנת משקל שונה לשגיאות שונות. למשל, שגיאה בין 1 ל- 8 גרועה יותר משגיאה בין 1 ל-4 (ספרות יחסית דומות). ראינו דוגמה לפונקציה γ כזו בה ההפרש בין המספרים משפיע על השגיאה. ראינו ש- w^1 הכיל פחות אלמנטים של הספרה 8 בלבן ו- w^8 הכיל פחות אלמנטים של הספרה 1 בלבן. אם מגדירים לדוגמה $\gamma(4,8)=\gamma(8,4)=0$ (לא אכפת לנו אם תהיה טעות בין 4 ל-8), אז נראה את 4 באופן ברור בווקטור של 8 ולהיפך. ראינו שכשהשתמשנו בפונקציית γ כזו שבה אנחנו מאחדים שתי מחלקות, זה מקטין את מס' השגיאות של ה- SVM.

מנועי חיפוש ובעיית הדירוג

הרעיון הכללי של מנוע חיפוש הוא לדרג מסמכים לפי השאילתה (Query).



ה-Crawler (שנקרא בעבר Spider) נכנס לדפי אינטרנט ומוריד את כל התוכן לשרתי מנוע החיפוש. למשל, ב- Google יש את כל האינטרנט בזיכרון כ-10 פעמים (כשלוחצים על Cached, זה מה שמקבלים).

ה-Parser מוריד את הסימנים של ה- html (לכן כשצופים ב- Cached page, הוא נראה פחות מעוצב) והוא גם בונה גרף לינקים (Web graph) של כל הדפים הנכנסים והיוצאים מהדף הנוכחי. בהמשך, לפי ה- Web graph, הוא "כורה" דף אחר שאליו הגיע דרך לינק.

ה- Indexer בונה שני סוגי אינדקסים: forward שמכיל לכל מסמך את המילים המופיעות בו, ו- inverted שמכיל לכל מילה את המסמכים בהם היא מופיעה. הרעיון הוא לקחת את כל המילים

שמופיעות בדף, כדי שכאשר מישהו יחפש את המילים האלה, נקבל בקלות הפנייה לדף הנוכחי. המוטיבציה באינדקסים היא לא לעבור על כל האינטרנט כדי לחפש את מילות חיפוש במסמכים.

ה- Link Analyzer מדרג דפים לפי מספר הקישורים שיש אליהם. ב- Cache יש תוצאות לשאילתות נפוצות (למשל פייסבוק).

<u>ווקטור המאפיינים (דירוג מסמכים)</u>
קיימות כמה דרכים לדרג מסמך לפי שאילתה. רוב מנועי החיפוש היום משתמשים בשילוב של כמה מדדים, שכל אחד מהם הוא מאפיין ב- Feature Vector.

עבור המדדים הבאים, בשאילתה, מתייחסים לכל מילה בנפרד, מוציאים את כל המסמכים המכילים את המילה הזו ומחשבים עבורה את המדד:

<u>TF</u> - $TF(t,d)=f(t,d)$ - התדירות של המילה t במסמך d, באופן מנורמל ביחס לשאר המילים במסמך (מה שמעיד על חשיבות המילה במסמך). פונקציה אפשרית היא לחלק את התדירות של t במסמך בתדירות של המילה עם התדירות המקסימלית במסמך. בזמן האימון t היא מילה במסמך ובזמן החיפוש t היא אחת ממילות השאילתה ו-d הוא כל אחד מהמסמכים המכילים את t (לפי האינדקס).

<u>IDF</u> – $IDF(t)=\log\left(\frac{N}{n(t)}\right)$ – מס' המסמכים המכילים את t ביחס לכל המסמכים. אם המילה נדירה, היא תקבל ציון גבוה, אחרת היא תקבל ציון נמוך. זה מאפשר לנטרל אפקט של מילים פונקציונליות כמו "את", "the", "to" וכו'.

<u>TF-IDF</u> – מכפלה של TF ו- IDF.
<u>BM25</u> – יוריסטיקה טובה שמשתמשת ב- TF ו- IDF:

$$BM25(d,q)=\sum_{i=1}^{M}\frac{IDF(t_i)\cdot TF(t_i,d)\cdot(k_1+1)}{TF(t_i,d)+k_1\cdot(1-b+b\cdot\frac{len(d)}{avdl})}$$

<u>LMIR</u> (Language Model for Information Retrieval) – מודל שפה שנותן הסתברות למילה בהינתן מסמך.

מדד אחר שלא מתייחס למילים בחיפוש אלא רק נותן ציון למסמך:

<u>Page Rank</u> – סוכמים לכל מסמך המצביע אל המסמך הנוכחי את היחס בין ה- PR שלו למס' הלינקים היוצאים שלו:

$$PR(d_u)=\sum_{d_u\in B_u}\frac{PR(d_v)}{U(d_v)}$$

אם המסמכים המצביעים למסמך הנוכחי הם חשובים והם לא מצביעים לעוד הרבה מסמכים, המסמך חשוב. ה- Crawler הולך הליכה רנדומית כמו גולש טיפוסי. α היא ההסתברות להמשיך להיכנס ללינקים ו- $(1-\alpha)$ היא ההסתברות לצאת ולהיכנס לדף אחר לגמרי. זה סוג של Backoff שמטפל בכך שיכול להיות שיש עוד דפים אבל הגולש עזב את הלינקים.

$$PR(d_u)=\alpha\sum_{d_u\in B_u}\frac{PR(d_v)}{U(d_v)}+\frac{1-\alpha}{N}$$

ה- Feature Vector של החיפוש יכיל את המדדים הנסמכים על TF-IDF על תוכן המסמך, ה-URL, הכותרת וכו', ה- PR ומדדים נוספים. יש הרבה כפילות במאפיינים כדי שחלק מהמאפיינים יקבלו ציון גבוה יותר (מאותה הסיבה שלא מספיק לקחת שני מאפיינים בבעיה כזו, גם אם הם מאוד אינדיקטיביים).

<u>פונקציית ה-Loss (ביצועים)</u>
לכל מסמך, נתנו למדרגים אנושיים לדרג את המסמכים לפי השאילתה והרלוונטיות שלהם מ-0 (לא רלוונטי בכלל) עד 4 (מאוד רלוונטי). חישבו את הציון של כל מסמך לפי ווקטור המאפיינים.
נרצה למצוא מדד המודד את איכות מנוע החיפוש. ניתן להשתמש במדדים הרגילים:

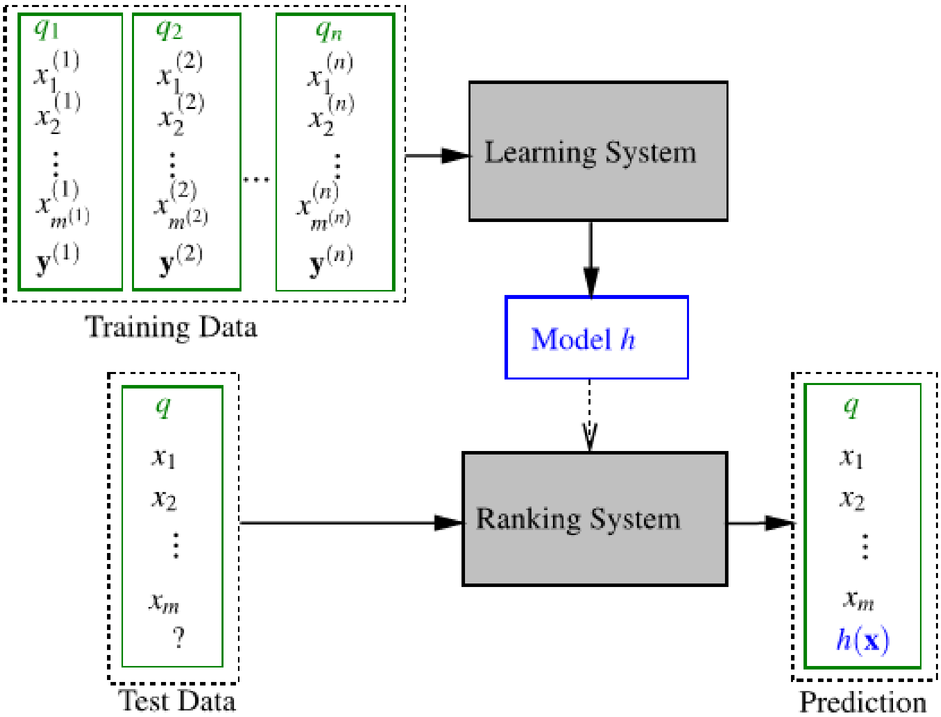
- <u>Precision</u> – אחוז המסמכים הרלוונטיים שחזרו ממנוע החיפוש מבין כל המסמכים שחזרו.
- <u>Recall</u> – אחוז המסמכים הרלוונטיים שחזרו ממנוע החיפוש מבין כל המסמכים הרלוונטיים.
- <u>F-measure</u> – ממוצע הרמוני של Precision ו- Recall.

הבעיה המרכזית עם מדדים אלה היא שהם לא מתייחסים לסדר הדירוג של המסמכים. חשוב שמסמכים רלוונטיים יוחזרו במקום גבוה (ולא רק שיוחזרו). המדדים הבאים מתייחסים לסדר:

- <u>Precision @ j</u> – לכל נקודת דירוג j, Precision במקום ה- j הוא מספר המסמכים הרלוונטיים שחזרו מתוך j המסמכים שחזרו.
- <u>Average Precision</u> - ממוצע של Precision @ j לכל נקודת דירוג j.
- <u>MAP (Mean Average Precision)</u> - ממוצע של AP עפ"י שאילתות רבות.
- <u>DCG (Discounted commutative gain)</u> – נותנים "עונש" שונה לטעות בכל נקודת דירוג. בנוסף, המדד הזה לוקח בחשבון את מידת הרלוונטיות של המסמך. למשל, אם טעינו בנקודת הדירוג הראשונה, העונש יהיה גדול יותר מאשר אם טעינו בנקודת דירוג נמוכה יותר, ואם לא הכללנו מסמך מאוד רלוונטי או שהכללנו מסמך מאוד לא רלוונטי, העונש יהיה גבוה יותר מאשר אם לא הכללנו מסמך שהוא קצת רלוונטי או שהכללנו מסמך שהוא לא כ"כ רלוונטי. מחשבים DCG לכל נקודת דירוג ומחברים אותם.

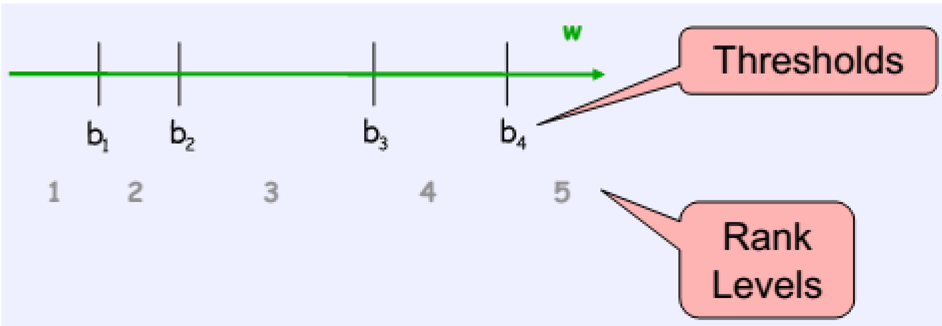
__אלגוריתמים לדירוג__

לכל דוגמת אימון יש שאילתה q_i, רשימת מסמכים $x_1^i,\dots,x_m^i$ ודירוג מסמכים y^i.



יש כמה שיטות עיקריות ללמידה, ונתאר לכל אחת מהן אלגוריתם אחד שמממש אותה.

- __Point wise__ – שיטה נקודתית – מקבלים שאילתה ומסמך ומדרגים עד כמה המסמך רלוונטי לשאילתה, לדוגמה בין 0 (לא רלוונטי) ל- 2 (מאוד רלוונטי). השיטה הכי פשוטה היא לעשות רגרסיה ולתת פרדיקציה ל- 0,1,2. אפשר להשתמש ב- SVM בינארי (רלוונטי או לא) או Multiclass (0,1,2). שיטה נוספת היא __Ordinal Regression-Base (PRank)__:



מכפילים את w ב- x. לומדים כמה ספים $b_1,\dots,b_4$ שקובעים את מידת הרלוונטיות (1,...,5) – כלומר, אם $b_1 < wx < b_2$ אז מידת הרלוונטיות היא 2. אם התוצאה הנכונה שונה (למשל 4), צריך לעדכן את w ואת b בהתאם (זה אלגוריתם דסקרימינטיבי). מגדירים את הספים שנכללים ב- Error Set (בדוגמה הזו: $E=\{b_2,b_3\}$) ומזיזים אותם אחורה ($b_r \leftarrow b_r - 1$), כדי להקטין את השגיאה (ההזזה יכולה להוציא אחד מהם מ-E). בנוסף, מזיזים את w בזווית כדי לקרב אותו לתוצאה האמיתית: $w \leftarrow w + |E|x$.

הדירוג של המסמכים הוא לפי מידת הרלוונטיות שלהם (דירוג חלש).

עבור k ספים, אם $\left\|x_i\right\|^2 \le R^2$ ו- $\left\|w\right\|^2 + b_1^2 + \cdots + b_{k-1}^2 = 1$ ו- $margin > 0$, אז מספר השגיאות של האלגוריתם חסום ע"י $(k-1)\frac{R^2+1}{margin^2}$.

- <u>Pair wise</u> – מקבלים שאילתה וזוג מסמכים u, v וקובעים מי מהמסמכים יותר רלוונטי. לבסוף מחזירים את הדירוג על כל המסמכים. דוגמה לאלגוריתם כזה היא <u>SVM-Rank</u>:
דוגמאות האימון הן:
$$\left\{\left(q^1, d_1^1, \dots, d_{n_q}^1, y_1^1, \dots, y_{n_q}^1\right), \dots, \left(q^m, d_1^m, \dots, d_{n_q}^m, y_1^m, \dots, y_{n_q}^m\right)\right\}$$
דוגמה מורכבת משאילתה, מסמכים ורלוונטיות (y_i^j – האם המסמך ה-j רלוונטי לשאילתה ה-i).
פרדיקציה - לכל דוגמת אימון, ממיינים את המסמכים לפי: $\hat{y} = argsort_{d_j^i} w \cdot \phi\left(q^i, d_j^i\right)$.
כאשר $\phi = TF - IDF$. נגדיר:
$D^{i+} = \left\{d_j^i \middle| y_j^i = +1\right\}$ קבוצת המסמכים הרלוונטיים.
$D^{i-} = \left\{d_j^i \middle| y_j^i = -1\right\}$ קבוצת המסמכים הלא-רלוונטיים.

$r = argmin_{d_j^i \in D^{i+}} w^t \cdot \phi\left(q^i, d_j^i\right)$ המסמך הרלוונטי שדורג הכי נמוך.
$s = argmax_{d_j^i \in D^{i-}} w^t \cdot \phi\left(q^i, d_j^i\right)$ המסמך הלא-רלוונטי שדורג הכי גבוה.

במקרה של שגיאה בדירוג, $w \cdot \phi(q, s) > w \cdot \phi(q, r)$.
פונקציית ה-Loss מוגדרת להיות: $loss = 1 - w \cdot \phi(q, r) + w \cdot \phi(q, s)$.
בפועל מגדירים: $l = \max\{0, 1 - w \cdot \phi(q, r) + w \cdot \phi(q, s)\}$.

במקרה של שגיאה, העדכון של w הוא: $w = w + \tau \cdot [\phi(q, r) - \phi(q, s)]$
עבור קבוע למידה τ. שלב האימון הוא (Passive-Aggressive):

for $i = 1, \dots, m$
 $r = argmin_{d_j^i \in D^{i+}} w^t \cdot \phi\left(q^i, d_j^i\right)$
 $s = argmax_{d_j^i \in D^{i-}} w^t \cdot \phi\left(q^i, d_j^i\right)$
 $l = \max\left\{0, 1 - w^t \cdot \phi\left(q^i, r\right) + w^t \cdot \phi\left(q^i, s\right)\right\}$
 if $(l > 0)$:
  $\tau = \dfrac{l}{\left|\left|\phi(q^i, r) - \phi(q^i, s)\right|\right|^2}$
  $w^{t+1} = w^t + \tau \cdot \left[\phi\left(q^i, r\right) - \phi\left(q^i, s\right)\right]$

בשלב ההסקה, מדרגים לפי: $\hat{y} = argsort_{d_j^i} w \cdot \phi\left(q^i, d_j^i\right)$

- <u>List wise</u> – מקבלים שאילתה ורשימת מסמכים ומדרגים אותם לפי רלוונטיות.
דוגמה לאלגוריתם כזה היא <u>SVM-MAP</u>: וריאציה של Structured SVM (בהמשך) שממקסמת ציון MAP (Mean Average Precision) על הדירוג של כל המסמכים בדוגמאות האימון.

<u>Input</u>: $(x_1, y_1), \dots, (x_m, y_m)$ where x_i is the query and documents and y_i is the set of binary labels for the documents relevance.
1. $w_0 = 0$
2. for $t = 1\ to\ t$
 2.1. choose (x_i, y_i) randomly
 2.2. compute $L = \max_{\hat{y}}[MAP(y_i, \hat{y}) - w \cdot \psi(x_i, y_i) + w \cdot \psi(x_i, \hat{y})]_+$
 2.3. if $L > \epsilon$
  2.3.1. $w_t = w_{t-1}(1 - \eta\lambda)w(1 - \eta\lambda) - \eta\left(\psi(x_i, \hat{y}_L) - \psi(x_i, y_i)\right)$
3. end

הפונקציה ψ מקבלת דירוג y ביחס למילות שאילתה ומסמכים x ומוגדרת באופן הבא:

$$\psi(x,y)=\frac{1}{|C^x||C^{\bar{x}}|}\sum_{i:d_i\in C^x}\sum_{j:d_j\in C^{\bar{x}}}\left[y_{ij}\left(\phi(x,d_i)-\phi(x,d_j)\right)\right]$$

כאשר C^x הם המסמכים הרלוונטיים (המסמכים שה- label האמיתי שלהם הוא 1),
ו- $C^{\bar{x}}$ הם המסמכים הלא-רלוונטיים (המסמכים שה- label האמיתי שלהם הוא 1-).
$y_{ij}=+1$ אם המסמך ה- i דורג לפני המסמך ה- j, אחרת $y_{ij}=-1$.
ϕ הוא ווקטור המאפיינים של המסמך ביחס לשאילתה. כלומר, $\psi(x,y)$ הוא ווקטור מאפיינים חדש שהוא סכום ההפרשים בין מסמכים רלוונטיים ללא-רלוונטיים.

המטרה היא למזער את תוחלת ה- MAP Loss, אבל כיוון שההתפלגות אינה נתונה, ממזערים את הממוצע של ה-Loss על דוגמאות האימון ומוסיפים גורם רגולריזציה:

$$w^*=argmin_w\frac{1}{m}\sum_{i=1}^{m}L\left(y_i,\hat{y}_w(x_i)\right)+\frac{\lambda}{2}\left||w|\right|^2$$

במקום למזער את הביטוי הזה, ממזערים את ה- Hinge Loss (חסם עליון):

$$w^*=argmin_w\left(\frac{1}{m}\sum_{i=1}^{m}L_{hinge}\left(y_i,\hat{y}_w(x_i)\right)\right)+\frac{\lambda}{2}\left||w|\right|^2$$

כאשר:

$$L_{hinge}\left(y_i,\hat{y}(x_i)\right)=\max_{\hat{y}}[MAP(y_i,\hat{y})-w\cdot\psi(x_i,y_i)+w\cdot\psi(x_i,\hat{y})]_+$$

מחזיר Loss ביחס למבנה הכי גרוע $\hat{y}$ שיכול להתקבל עבור הקלט x_i (מפר הכי הרבה אילוצים):

$$w^*=\frac{\lambda}{2}\left||w|\right|^2+\frac{1}{n}\sum_{i=1}^{n}\max_{\hat{y}}[MAP(y_i,\hat{y})-w\cdot\psi(x_i,y_i)+w\cdot\psi(x_i,\hat{y})]_+$$

לצורך חישוב ה- Hinge Loss, צריך למצוא את y הכי גרוע (שורה 2.2).

כלל העדכון של w (שורה 2.4) נובע מ- SGD:

$$w=w-\eta\nabla_w\left[MAP(y_i,\hat{y}_L)-w\cdot\psi(x_i,y_i)+w\cdot\psi(x_i,\hat{y}_L)+\frac{\lambda}{2}\left||w|\right|^2\right]=$$
$$=w-\eta(-\psi(x_i,y_i)+\psi(x_i,\hat{y}_L)+\lambda w)=w(1-\eta\lambda)-\eta\left(\psi(x_i,\hat{y}_L)-\psi(x_i,y_i)\right)$$

הפרדיקציה היא: $\hat{y}=argmax_{y\in Y}w\cdot\psi(x,y)$, כלומר בוחרים את הדירוג שעבורו סכום ההפרשים בדירוג בין המסמכים הרלוונטיים ללא-רלוונטיים יהיה מקסימלי.

Structure Prediction

הגדרת הבעיה

כל אחת מהבעיות שלמדנו (בינארית, Multi-class, Multi-label) מכלילה את הבעיה הקודמת. הבעיה הכללית ביותר היא Structure Prediction, שזו בעיה מורכבת יותר שקשה להמיר אותה לבעיה בינארית. בבעיה הזו צריך לנבא מבנה, כמו בדוגמאות הבאות:

- OCR – בבעיית ה- OCR שממומשת כ- Multiclass, לא מנצלים את ההקשר: כל אות מזוהה באופן בלתי תלוי. היינו יכולים להשתמש באותיות הקודמות ולזהות את הרצף.
- CFG Parsing – המטרה היא לבנות את עץ התחביר של המשפט.
- Bilingual Word Alignment – למצוא את ה- Alignment בין שני משפטים.
- POS Tagging – מחפשים רצף POS.

פתרון בעיית Structure Prediction

הרעיון הוא לבנות מסווג שנותן החלטה מסוג של סדרה, עץ או מבנה. יש כמה בעיות עם זה:

- הקלט והפלט לא בהכרח באורך קבוע. למשל זוג משפטים ו-alignment.
- מרחב החיפוש של הפלט הוא לפעמים אקספוננציאלי. למשל ב- alignment, יש מס' אפשרויות לבחור זוגות ולפעמים יש גם סגמנטציה ל- Phrases.
- איך לבדוק את ההצלחה – evaluation metric / measure of performance. בסיווג בינארי, בדקנו כישלון ע"י 0-1-loss: $\delta(\hat{y} \neq y)$. במבנה, נרצה לבדוק כמה המבנה שונה מהמבנה האמיתי, כלומר המדד צריך להיות פרופורציונלי לשינוי. למשל, ב- MT המדד הוא BLEU, בזיהוי דיבור המדד הוא WER (Word Error Rate) – מרחק עריכה בין סדרות המילים, וכו'.

הגדרה פורמלית

נתונות דוגמאות אימון $X = (x_1, \dots, x_m) \in X^m$ עם מבנה אמיתי $Y = (y_1, \dots, y_m) \in Y^m$.

פונקציית הציון היא $X \times Y \to R$, כלומר היא נותנת ציון למבנה Y שהתקבל עבור קלט X.

הפרדיקציה היא המבנה שממקסם את הציון עם הקלט:

$$\hat{y}_w = argmax_y w \cdot \phi(x, y)$$

כאשר הפונקציה ϕ נקראת Feature Map ומוגדרת $\phi: X \times Y \to R^d$.

פונקציית ה-Loss: $L(\hat{y}, y)$ מוגדרת להיות המרחק בין הפרדיקציה לבין המבנה האמיתי.

המטרה היא להביא למינימום את תוחלת ה-Loss, כלומר:

$$w^* = argmin_w E[L(y, \hat{y}_w)]$$

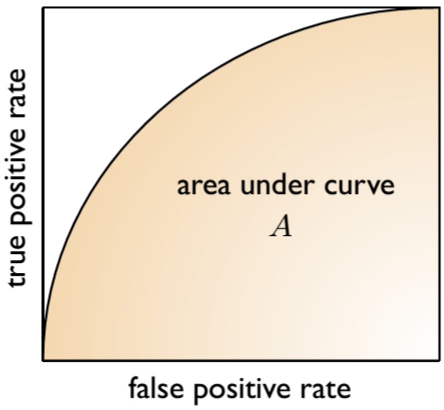
לפעמים מדברים על מקסימיזציה כאשר המדד הוא של דמיון ולא של שוני (כמו BLEU ב-MT).

פונקציית AUC – Loss

פונקציית למדד חיזוי במקרה של זיהוי אירועים (למשל, בדוגמה בהמשך: זיהוי מופע של מילה בדיבור). יהי $f: X \to R$ מסווג שעבור $x \in X$ מחזיר ערך ממשי, כך שאם $f(x) > c$, עבור c סף כלשהו, המסווג קובע כי המופע מכיל את האירוע, אחרת לא. נסמן ב-x^+ מופע המכיל אירוע, וב- x^- מופע שאינו מכיל אירוע.

TP - True Positive: מופע המכיל אירוע והמסווג זיהה בו אירוע: $f(x^+) > c$.
FP - False Positive: מופע שאינו מכיל אירוע, אך המסווג זיהה בו אירוע: $f(x^-) > c$.
TN - True Negative: מופע שאינו מכיל אירוע והמסווג לא זיהה בו אירוע: $f(x^-) < c$.
FN - False Negative: מופע המכיל אירוע, אך המסווג לא זיהה בו אירוע: $f(x^+) < c$.

גרף ROC (Receiver Operator Characteristic) הוא גרף של True Positive Rate ($\frac{TP}{TP+FN}$) לעומת False Positive Rate ($\frac{FP}{FP+TN}$):



פונקציית ה-Loss היא השטח מתחת לגרף, שנקרא AUC (Area under curve), ומוגדר כאינטגרל על מכפלת המדדים:

$$\begin{aligned} A &= \int_0^1 P(f(\boldsymbol{x}^+) > c)\, dP(f(\boldsymbol{x}^-) > c) \\ &= \int_{-\infty}^{+\infty} P(f(\boldsymbol{x}^+) > c)\, p_{f(\boldsymbol{x}^-)}(c)\, dc \\ &= P(f(\boldsymbol{x}^+) > f(\boldsymbol{x}^-)) \end{aligned}$$

כאשר $P_{f(x^-)}(c)$ היא פונקציית צפיפות ההסתברות של משתנה x^- בנקודה c.

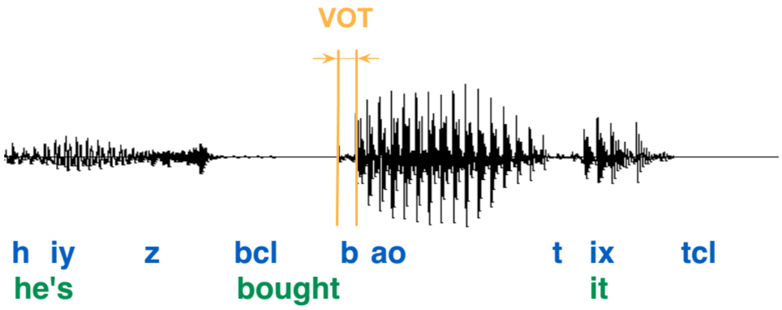
ניתן לראות שלמעשה האינטגרל מבטא את ההסתברות לכך שעפ"י המסווג, מופע חיובי יקבל ציון גבוה יותר ממופע שלילי. השטח הטוב ביותר הוא 1 (מופע חיובי תמיד מקבל ציון טוב יותר ממופע שלילי), והגרוע ביותר הוא 0.5 (אם הוא פחות מחצי, אז הופכים את הניבוי). כלומר, המטרה היא למקסם את השטח.

ניתן להשתמש ב-AUC כמדד להצלחה של מסווג, והמטרה היא למקסם אותו.

דוגמאות לשימוש ב-Structure Prediction

דוגמה #1 – זיהוי דיבור

מחקרים שמשווים בין סגנונות דיבור (קול) של אנשים שונים. מחקר משנת 2000 עוקב אחרי קולה של מלכת אנגליה בנאומה כל שנה. הוא גילה שהקול שלה מתקרב לדיבור עממי. במחקר אחר, בדקו אם תוך דקות, שני אנשים שמדברים מתכנסים לאותו סגנון דיבור. מחקר בטווח של שבועות בוצע על שידורי האח הגדול מאנגליה ב-2008. ניסו להראות שאנשים ששונאים זה את זה ידברו שונה ואנשים שאוהבים זה את זה ידברו דומה. סגנון הדיבור נבחן ע"י ה-VOT – Voice Onset Time – הזמן בין העיצור לבין הרעד של מיתרי הקול:



ייצוג כבעיית Structure Prediction:

<u>קלט</u> - $x = (x_1, \dots, x_T)$: סדרת ווקטורים של סיגנלים אקוסטיים.

<u>פלט</u> - $\hat{y} = \hat{t_v} - \hat{t_b}$ – ה- VOT: הזמן בין ה- burst (העיצור) לבין ההתחלה של הרעד במיתרי הקול.

<u>פונקציית ה- loss</u> - $L(\hat{y}, y) = \max\{|y - \hat{y}| - \epsilon, 0\}$ – אם ההפרש ביניהם קטן מאפסילון, זה נחשב נכון. $y = t_v - t_b$ כלומר הזמן הנכון. המטרה היא למצוא w כך שתוחלת ההפסד $E[L(\hat{y}, y)]$ מינימלית. הפרדיקציה היא: $\hat{y} = argmax_y w \cdot \phi(x, y)$.

מאפיינים ב-ϕ כוללים למשל $\phi_i(x, y) = N(y; \mu_{VOT}, \Sigma_{VOT})$: ההסתברות ל- VOT בהינתן ההתפלגות על האורך הטיפוסי, מרחקים בין אנרגיה נמוכה לגבוהה בזמנים של y וכו'.

דוגמה #2 – זיהוי מילות מפתח בדיבור

למשל, עבור האזנות לשיחות טלפון. רוצים לזהות מילים שעבורן יאזינו לשיחה. מחקר דומה רצה לזהות אוטיזם בגיל 24-48 חודשים. מאפיין לחשד לאוטיזם זה שאם מדברים עם הילד על מקום מסוים, הוא לא יסתכל לשם. שמו לילדים מצלמה על הכובע ועקבו אחרי העיניים שלהם בשיחה. שתי מערכות פעלו במקביל: זיהוי מילות מפתח (למשל "אחרת" – המורה דיברה עם הילד על הכיתה האחרת) ומעקב אחרי העיניים. המטרה היא לזהות באחוזי הצלחה מאוד גבוהים את זמן ההתחלה והסיום של מילה מסוימת בשיחה. ככל שמכניסים יותר מילים שרוצים למצוא, האיכות יורדת והזמן עולה.

ייצוג כבעיית Structure Prediction:

<u>קלט</u> - $x = (x_1, \dots, x_T)$: סדרת ווקטורים של סיגנלים אקוסטיים ומילה k שיש לזהות.

<u>פלט</u> – האם k נמצאה ובאיזה זמן התחלה וסיום. $\max_t f(x, k, t) > 0 \Rightarrow yes$ כאשר t הוא הזמן שהכי סביר שהמילה הופיעה.

פונקציית ה-Loss היא מדד AUC, על מופעים של המילה:

$$A = \mathbb{P}\left[\max_{\bar{t}} f_{\mathbf{w}}(\bar{\mathbf{x}}^{+}, k, \bar{t}) > \max_{\bar{t}} f_{\mathbf{w}}(\bar{\mathbf{x}}^{-}, k, \bar{t})\right]$$

כלומר, רוצים למקסם את ההסתברות ש-f_w תתן ציון גבוה יותר למופע חיובי של המילה k ב-x^+ בזמן t, מאשר למופע שלילי x^- שאינו מכיל את המילה k.

רוצים למצוא w הממקסם את A. יותר קל להביא למינימום את תוחלת מס' הטעויות (מס' הפעמים שמופע חיובי קיבל ציון נמוך יותר ממופע שלילי) – זה 0-1 Loss. מחליפים את התוחלת בממוצע על דוגמאות האימון ומוסיפים גורם רגולריזציה:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{m}\sum_{i=1}^{m} \delta\left[\max_{\bar{t}} f_{\mathbf{w}}(\bar{\mathbf{x}}_i^{+}, k_i, \bar{t}) < \max_{\bar{t}} f_{\mathbf{w}}(\bar{\mathbf{x}}_i^{-}, k_i, \bar{t})\right] \;+\; \frac{\lambda}{2}\|\mathbf{w}\|^2$$

במקום 0-1 Loss, משתמשים בחסם העליון, ובסה"כ מקבלים:

$$\boxed{\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{m}\sum_{i=1}^{m} \left[1 - f_{\mathbf{w}}(\bar{\mathbf{x}}_i^{+}, k_i, \bar{t}_i^{+}) + \max_{\bar{t}} f_{\mathbf{w}}(\bar{\mathbf{x}}_i^{-}, k_i, \bar{t})\right]_+ \;+\; \frac{\lambda}{2}\|\mathbf{w}\|^2}$$

כאשר מתקבל $\text{margin} = 1$.

הפרדיקציה היא: $\hat{y} = argmax_y w \cdot \phi(x,p,y)$ כאשר p הוא אוסף פונמות של המילה ה- k, y הוא אוסף הפונמות שזוהה ו-x הוא אוסף סיגנלי הקלט.

מאפיינים ב-ϕ כוללים למשל את סכום המרחקים בין הפונמות. בפועל מחזירים את הזמן של כל פונמה במילה ה- k. המרחק בין הפונמות יהיה קטן אם המילה נאמרה בשיחה וגדול אחרת (למשל, אם הפונמות הן חלק ממילים אחרות). מאפיין אחר יכול להיות סכום ה- confidence בזיהוי כל פונמה במילה (פלט של מסווג פונמי Multiclass). בנוסף, אפשר למדוד אורך טיפוסי של פונמות ולסכום על כל הפונמות המילה את ההסתברות לכל פונמה עפ"י האורך: $\sum_p N(x,p;\mu_p,\Sigma_p)$.

פתרונות לבעיית Structure Prediction

Structured Perceptron

המימוש הראשון. השתמשו בו ב- POS Tagging. האלגוריתם:

```
w_0 = 0
S = {(x_i, y_i)}
for t = 1 to T
        choose (x_i, y_i) randomly
        predict ŷ_i = argmax_y w · φ(x_i, y)
        if (ŷ_i ≠ y_i)
                update w: w_t = w_{t-1} + φ(x, y_i) − φ(x, ŷ_i)
        end
end
```

כלל העדכון: $w = w + \phi(x_i, y_i) - \phi(x_i, \hat{y}_i)$, כלומר במקרה של טעות, מורידים את ווקטור המאפיינים של הניבוי ומוסיפים את ווקטור המאפיינים של המבנה הנכון.
נשים לב שהאילוץ שפרספטרון מקיים הוא שלכל דוגמת אימון, הניבוי הנכון יהיה גבוה מכל השאר:

$$\hat{y}_i = argmax_y w \cdot \phi(x_i, y) \Rightarrow \forall y \neq \hat{y}_i : w \cdot \phi(x_i, \hat{y}_i) - w \cdot \phi(x_i, y) > 0$$

פרספטרון מקיים רק אילוצים, ובהמשך נראה אלגוריתמים אחרים שמנסים בנוסף למזער את L.

Structured SVM

המטרה היא למזער את תוחלת פונקציית ה- Loss (המדד, למשל WER), אבל כיוון שההתפלגות D אינה נתונה, ממזערים את הממוצע של ה-Loss על דוגמאות האימון ומוסיפים גורם רגולריזציה:

$$w^* = argmin_w E_{(x,y)\sim D}\left[L\big(y, \hat{y}_w(x)\big)\right] = argmin_w \frac{1}{m}\sum_{i=1}^{m} L\big(y_i, \hat{y}_w(x_i)\big) + \frac{\lambda}{2}\left||w|\right|^2$$

ראינו שלא תמיד ניתן למזער את L עצמה ולכן בסיווג בינארי, מצאנו חסם עליון (Hinge Loss): $l_{hinge}(y, z) = \max(0, 1 - yz)$, שמזעורו ממזער גם את L. ב-Structure Prediction, החסם העליון הוא: $l_{hinge}^{\Delta}\big(f, (x, y)\big) = \max_{z \in y}[\Delta(z, y) + f(x, z) - f(x, y)]$.

הוכחה:

$$\Delta(\hat{y}, y) = \Delta(\hat{y}, y) + w\phi(x, y) - w\phi(x, y) \leq \max_{\hat{y}_z}[\Delta(y, \hat{y}_z) + w\phi(x, \hat{y}_z)] - w\phi(x, y) \leq$$
$$\max_{z \in y}[\Delta(z, y) + f(x, z) - f(x, y)] = l_{hinge}^{\Delta}\big(f, (x, y)\big)$$

כלומר, בסה"כ יש למצוא: $w^* = argmin_w \left(\frac{1}{m}\sum_{i=1}^{m} L_{hinge}\big(y_i, \hat{y}_w(x_i)\big)\right) + \frac{\lambda}{2}\left||w|\right|^2$

$$L_{hinge}\big(y_i, \hat{y}(x_i)\big) = \max_{\hat{y}}[L(y_i, \hat{y}) - w \cdot \phi(x_i, y_i) + w \cdot \phi(x_i, \hat{y})]$$

קל לראות (מההוכחה הנ"ל) ש- $L\big(y_i, \hat{y}_w(x_i)\big) \leq L_{hinge}\big(y_i, \hat{y}(x_i)\big)$.
ל-Hinge Loss אין משמעות ביחס לניבוי. הוא מחזיר ציון ביחס למבנה הכי גרוע שיכול להתקבל עבור הקלט בהינתן ווקטור המשקלות הנוכחי. הרעיון הוא שאם נמזער אותו לכל דוגמאות האימון, נגיע ל-w שציון ה- Loss שלו לניבוי יהיה נמוך יחסית. לצורך חישוב ה-Hinge Loss, צריך למצוא את y הכי גרוע (שמפר את האילוצים הכי הרבה): $\hat{y}_L = argmax_{\hat{y}} w \cdot \phi(x_i, \hat{y}) + L(y_i, \hat{y})$. עושים זאת ע"י גזירה והשוואה ל-0. לכן, מציאת $\hat{y}_L$ זו בעיה שקיום פתרון עבורה תלוי ב- L, האם הפונקציה קמורה או גזירה. בסופו של דבר מקבלים שצריך למזער מדד Hamming בין $\hat{y}_L$ ל- y_i (ללא הוכחה).

כלל העדכון של w הוא: $w_t = w_{t-1} - \eta\nabla_w(L) = w_{t-1}(1 - \eta\lambda) + \phi(x_i, y_i) - \phi(x_i, \hat{y}_L)$.
עם קבוע למידה $\eta = 1$, נקבל כלל עדכון: $w_t = w_{t-1} + \phi(x_i, y_i) - \phi(x_i, \hat{y}_L) - \lambda w_{t-1}$.

מזעור w לא באמת ממזער את L.
בשלב ההסקה, בוחרים את y שממקסם את הציון: $\hat{y} = argmax_y w \cdot \phi(x_i, y)$.

<u>CRF (Conditional Random Fields)</u>
הרחבה של Logistic Regression עבור בעיית Structure Prediction, ושיטה מאוד נפוצה.
המטרה היא למזער את תוחלת פונקציית ה- Loss ביחס להתפלגות האמיתית $\rho(y|x)$:

$$w^* = argmin_w E_{(x,y)\sim\rho(y|x)}[L(y, \hat{y}_w)]$$

כיוון שההתפלגות האמיתית $\rho(y|x)$ אינה נתונה, ממזערים את ה- log-likelihood השלילי על דוגמאות האימון ומוסיפים רגולריזציה:

$$w^* = argmin_w \frac{1}{m}\sum_{i=1}^{m} -ln\big(P_w(y_i|x_i)\big) + \frac{\lambda}{2}||w||^2$$

כאשר ההסתברות מוגדרת ע"י הציון של מבנה y מנורמל ע"י ציון של כל מבנה y' אחר:

$$P_w(y|x) = \frac{1}{Z}\exp\big(w \cdot \phi(x,y)\big) = \frac{e^{w\phi(x,y)}}{\sum_{y'\in y} e^{w\phi(x,y')}}$$

כלל העדכון של w מחושב ע"י GD: $w_t = w_{t-1} - \eta\nabla_w\left[-ln\big(P_w(y|x)\big) + \frac{\lambda}{2}||w||^2\right]$.

$$\nabla_w\left[-ln\big(P_w(y|x)\big) + \frac{\lambda}{2}||w||^2\right] = \nabla_w\left[-ln\left(\frac{e^{w\phi(x,y)}}{\sum_{y'\in y} e^{w\phi(x,y')}}\right) + \frac{\lambda}{2}||w||^2\right] =$$

$$= \nabla_w\left[-w\cdot\phi(x,y) + ln\sum_{y'\in y} e^{w\phi(x,y')} + \frac{\lambda}{2}||w||^2\right] = -\phi(x,y) + \frac{\sum_{y'\in y}\phi(x,y')e^{w\phi(x,y')}}{\sum_{y'\in y} e^{w\phi(x,y')}} + \lambda w =$$

$$= -\phi(x,y) + \sum_{y'\in y}\phi(x,y')\,P_w(y'|x) + \lambda w = -\phi(x,y) + E_{y|x}\phi(x,y) + \lambda w$$

כלל העדכון:

$$w = w - \eta\big(-\phi(x_i,y_i) + E_{y|x_i}\phi(x_i,y) + \lambda w\big) = w(1-\eta\lambda) + \eta\left(\phi(x_i,y_i) - E_{y|x_i}\phi(x_i,y)\right)$$

כלומר, עבור טעות בדוגמת אימון, מוסיפים את הציון של המבנה האמיתי y_i ומורידים את התוחלת של הציון לכל פרדיקציה אחרת.

גם כאן מזעור w לא מתייחס כלל ל-L. עם זאת, אם w נלמד כפי שמתואר, הוא מגדיר התפלגות $P_w(y|x)$. ניתן בשלב ההסקה להתייחס לפונקציית ה-Loss, ע"י בחירת המבנה שממזער את תוחלת ה-Loss, כלומר: $\hat{y}_w(x) = argmin_{\hat{y}} E_{P_w(y|x)}\big[L\big(y, \hat{y}(x)\big)\big]$ (במקום לבחור את y שממקסם את הציון: $\hat{y} = argmax_y w \cdot \phi(x_i, y)$).
במקרה זה, אם ההסתברות המוגדרת ע"י המודל שווה להסתברות שממנה נלקחו הדגימות, כלומר $P_w(y|x) = \rho(y|x)$ ונניח שפונקציית ה-Loss מוגדרת כ-Loss 0-1, כלומר $L(y, \hat{y}_w) = 1[y \neq \hat{y}_w]$, אז יתקיים:

$$\hat{y}_w(x) = argmin_{y'} E_{y|x}[1[y \neq \hat{y}_w]] = argmin_{y'} P(y \neq y'|x) = argmax_{y'} P(y = y'|x)$$

כלומר, בשלב ההסקה בוחרים את המבנה האופטימלי וממזערים נכון את ה- Loss.
עם זאת, מכיוון שברוב המקרים לא ניתן להניח כי $P_w(y|x) = \rho(y|x)$, אז קיימת חוסר עקביות מכיוון שמזעור w לפי הנוסחה: $w^* = argmin_w \frac{1}{m}\sum_{i=1}^{m} -ln\big(P_w(y_i|x_i)\big) + \frac{\lambda}{2}||w||^2$ ייתן תוצאה שונה מאשר מזעור לפי תוחלת ה-Loss.

Recommender Systems

מערכות שממליצות למשתמש על דברים אחרים על סמך הבחירות שלו. למשל Amazon מציעה ספרים אחרים על סמך הספרים שהמשתמש קנה, Pandora ממליצה על שירים דומים לשירים שהמשתמש בחר וכו'.

הדוגמה המנחה היא מערכת להמלצה על סרטים. נתונה מטריצה של סרטים ומשתמשים. כל אחד מהמשתמשים יכול לדרג את הסרטים שהוא ראה. עבור סרטים שהוא לא ראה מופיע סימן שאלה. המטרה היא לחזות את ערכי סימני השאלה לפי הדירוג של משתמשים אחרים לאותו הסרט ושל אותו המשתמש לסרטים אחרים.

	Alice	Bob	Carol	David
Love at last	5	5	1	1
Romance forever	5	?	?	1
Cute puppies of love	?	4	1	?
nonstop car crashes	1	1	5	4
Sword vs. karate	1	1	5	?

<u>מערכות המלצה מבוססות מאפיינים (Feature-based)</u>

הלמידה הכי בסיסית. לסרט ה-i יש ווקטור מאפיינים x^i. לדוגמה, ווקטור מאפיינים באורך 2 שבו המאפיין הראשון קובע הוא עד כמה הסרט הזה הוא סרט רומנטי, והשני עד כמה הוא סרט מתח.

	Alice	Bob	Carol	David	x_1	x_2
Love at last	5	5	1	1	0.9	0
Romance forever	5	?	?	1	1.0	0.01
Cute puppies of love	?	4	1	?	0.99	0.02
nonstop car crashes	1	1	5	4	0.1	1.0
Sword vs. karate	1	1	5	?	0.2	0.9

למשתמש ה-j יש ווקטור משקלות w^j שנלמד ע"י ווקטורי הסרטים והדירוגים.
$w^j x^i$ הוא הדירוג הצפוי של המשתמש j עבור הסרט i.

מסמנים $r(i,j) = 1$ אם המשתמש j צפה בסרט ה-i (0 אחרת), ו- $y^{i,j}$ הדירוג של j לסרט ה-i (אם מוגדר). הדירוג הצפוי הוא $w^j x^i$. מס' הסרטים שדורגו ע"י משתמש j הוא m^j.
את ווקטורי המשקלות של המשתמשים לומדים ע"י ווקטורי הסרטים והדירוגים הקיימים:

$$w^j = \min_{w^j} \frac{1}{m^j} \sum_{i:r(i,j)=1} \left(w^j \cdot x^i - y^{i,j}\right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n} \left|\left|w^j\right|\right|^2$$

גורם הרגולריזציה הוא: $\frac{\lambda}{2}\sum_{j=1}^{n}\left|\left|w^j\right|\right|^2$ על כל ווקטורי המשתמשים.

המטרה היא להביא למינימום את $\frac{1}{m^j}\sum_{i:r(i,j)}\left(w^j \cdot x^i - y^{i,j}\right)^2$, כלומר ממוצע ריבוע ההפרשים בין הדירוג הצפוי לדירוג בפועל לכל הסרטים שהמשתמש j צפה בהם.

כדי למצוא את ווקטורי כל המשתמשים, ממזערים את הסכום הנ"ל:

$$w = \min_{w^1,\dots,w^n} \sum_{j=1}^{n} \frac{1}{m^j} \sum_{i:r(i,j)=1} \left(w^j \cdot x^i - y^{i,j}\right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n} \left|\left|w^j\right|\right|^2$$

מעדכנים לפי GD:

$$w^j = w^j - \eta \frac{\partial w^j}{\partial f} = w^j - \eta \left(\lambda w^j + \frac{2}{m^j} \sum_{i:r(i,j)=1} \left(w^j \cdot x^i - y^{i,j}\right) x^i \right) \approx$$
$$w^j(1 - \eta\lambda) - \eta \sum_{i:r(i,j)=1} \left(w^j \cdot x^i - y^{i,j}\right) x^i$$

<u>פירוק מטריצה בדרגה נמוכה (Low-Rank Matrix Factorization)</u>

נגדיר את מטריצת הדירוגים בתור Y:

$$Y = \begin{bmatrix} 5 & 5 & 1 & 1 \\ 5 & ? & ? & 1 \\ ? & 4 & 1 & ? \\ 1 & 1 & 5 & 4 \\ 1 & 1 & 5 & ? \end{bmatrix}$$

X היא מטריצת הסרטים (השורה ה-i היא הווקטור של הסרט ה-i, x^i), ו-W היא מטריצת המשתמשים (השורה ה-j היא הווקטור של המשתמש ה-j, w^j):

$$X = \begin{bmatrix} -x^1- \\ -x^2- \\ \vdots \\ -x^l- \end{bmatrix} \qquad W = \begin{bmatrix} -w^1- \\ -w^2- \\ \vdots \\ -w^n- \end{bmatrix}$$

המטרה היא לגרום ל-$X \cdot W^T$ להיות דומה ככל האפשר ל-Y, כלומר, שהדירוגים הצפויים יהיו דומים למטריצת הדירוגים האמיתית:

$$\begin{bmatrix} w^1 \cdot x^1 & w^2 \cdot x^1 & \dots & w^n \cdot x^1 \\ w^1 \cdot x^2 & w^2 \cdot x^2 & \dots & w^n \cdot x^2 \\ \vdots & \vdots & \ddots & \vdots \\ w^1 \cdot x^l & w^2 \cdot x^l & \dots & w^n \cdot x^l \end{bmatrix}$$

בעיית האופטימיזציה משערכת את X,W לפי הנוסחה:

$$argmin_{X,W} \frac{1}{2} \left|\left|X \cdot W^T - Y\right|\right|_F^2, \qquad s.t. rank(X \cdot W^T) = r$$

ניזכר בהגדרת דרגת המטריצה: $rank(A) = \min\{\rho_C(A), \rho_R(A)\}$, כאשר $\rho_C(A)$ הוא מימד מרחב העמודות ו-$\rho_R(A)$ הוא מימד מרחב השורות.

במתמטיקה, Low-Rank Matrix Factorization היא בעיית אופטימיזציה, שבה פונקציית העלות מודדת את ההתאמה בין מטריצת נתונה (Y) ומטריצת קירוב ($X \cdot W^T$), בכפוף למגבלה שמטריצת הקירוב היא בעלת דרגה נמוכה. אילוץ הדרגה קשור למגבלה על המורכבות של דגם שמתאים לנתונים.

פונקציית ה"עלות" היא נורמת פורביניוס:

$$\left|\left|X \cdot W^T - Y\right|\right|_F^2 = \sum_{i=1}^{l} \sum_{j=1}^{n} \left|X_{ij} W_{ij} - Y_{ij}\right|^2$$

בפועל, רצים רק על סרטים שדורגו: $\left|\left|X \cdot W^T - Y\right|\right|_F^2 = \sum_{i,j:r(i,j)=1} \left|X_{ij} W_{ij} - Y_{ij}\right|^2$.

כלומר, המטרה היא למצוא את X, W שעבורם נורמת פורביניוס של ההפרש $X \cdot W^T - Y$ היא מינימלית, וכמו כן, הדרגה של $X \cdot W^T$ מוגבלת ל-r.

הפתרון מתבצע ע"י SVD (Singular Value Decomposition) של המטריצה Y:

$$\hat{Y} = X \cdot \hat{\Sigma} \cdot W^T$$

כאשר $\hat{\Sigma}$ היא מטריצת הערכים העצמיים של Y:

$$\hat{\Sigma} = \begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \end{pmatrix} \in R^{r,r}$$

$X \in R^{l,r}$ ו-$W^T \in R^{r,n}$, כך שבסה"כ, $Y \in R^{l,n}$ כנדרש.

אם רוצים להוסיף margin, פותרים:

$$\frac{1}{2} \left|\left|X\right|\right|_F^2 + \frac{1}{2} \left|\left|W\right|\right|_F^2 + \sum\nolimits_i \max\{0, 1 - Y \cdot X \cdot W^T\}$$