סיכום למידה וניתוח של מידע רב

הגדרה: **למידה מונחית** היא כזו שהlearner לומד ע"י סט של דוגמאות עם התווית שלהם. וה-learner צריך להחזיר כפלט היפותזה (פונקציה שממפה כל דוגמה אפשרית לתווית)
ההצלחה של אלגוריתם למידה כזה נמדד על פי הצלחתו לחזות נכונה תוויות של דוגמאות בעתיד.
לאלגוריתם כזה יש שני שלבים :
1. שלב האימון: בשלב זה ה-learner מקבל סט של דוגמאות והתוויות שלהם ומחזיר היפותזה .
2. שלב הטסט: אנו משתמשים בהיפותזה על דוגמאות חדשות ורואים את מידת ההצלחה של ההיפותזה.

פורמאלי :

- $\mathcal{X}$ - the set of all possible examples
- $\mathcal{Y}$ - the set of all possible labels
- A training sample: $S = ((x_1, y_1), \ldots, (x_m, y_m))$
- **Note**: S is a **sequence**: it has an order and can have **duplicates**.
- A **learning algorithm** is any algorithm that has:
  - **Input:** A training sample S
  - **Output:** A prediction rule $\hat{h}_S : \mathcal{X} \to \mathcal{Y}$.

אלגוריתם ה-Memorize

Memorize algorithm

input A training sample S
output A function $\hat{h}_S : \mathcal{X} \to \mathcal{Y}$.
1: Set $\hat{h}_S = f_S^{\text{mem}}$ where f_S^{mem} is defined as:

$$\forall x \in \mathcal{X}, f_S^{\text{mem}}(x) = \begin{cases} y & y \text{ is the first such that } (x, y) \in S \\ \text{a random label} & \text{otherwise} \end{cases}$$

In the second case, the label is drawn **uniformly at random** from $\mathcal{Y}$.

שימו לב: האלגוריתם לא לומד שום דבר, הוא סך הכל "משנן" מה התווית של כל דוגמה .
האלגוריתם הזה יהיה מוצלח אם הדוגמאות שנראה בעתיד יהיו ברובן כמו הדוגמאות שראינו כבר.
מסקנה : יש התפלגויות שעבורן זה אלגוריתם טוב ויש התפלגויות שעבורן הוא גרוע .

הגדרה: Training sample היא רשימה סדורה S כך שלכל דוגמה מוצמדת התווית שלה והיא נדגמת מתוך ההתפלגות D . $S = ((x_1, y_1) \ldots . (x_m, y_m)) | S \sim \mathcal{D}^m$.

מדידת ההצלחה של האלגוריתם:

הגדרה: הטעות של האלגוריתם מוגדרת להיות ההסתברות של ההיפותזה ליפול על תווית לא נכונה ביחס להתפלגות . פורמאלי :

$$err(\widehat{h_s}, \mathcal{D}) = P_{(X,Y)\sim\mathcal{D}}[\widehat{h_s}(X) \neq Y]$$

הגדרה: **החלטת התווית בצורה דטרמיניסטית בהינתן הדוגמה** מוגדר כך :

$$\forall x \in X, \exists y \; s.t: \; P[Y = y | X = x] = 1 \; or$$
$$\forall x \in X, there \; is \; only \; one \; y \in Y \; s.t: \mathcal{D}(x, y) > 0$$

הטעות של אלגוריתם ה- *Memorize* :

נניח שההתפלגות היא בעלת תכונת – החלטת התווית בצורה דטרמיניסטית בהינתן הדוגמה

ונניח ויש k תוויות , כלומר $|Y| = k$. נקבל כי הטעות היא $err(\widehat{h_s}, \mathcal{D}) = \frac{k-1}{k} M_s$ כאשר

$$M_s = \Sigma_{x \in X \setminus X_s} p_x \mid p_x = P_{(X,Y)\sim\mathcal{D}}[X = x] and\ X_s = \{x \mid \exists y\ s.t : (x,y) \in S\}$$

על מנת להבין מה הטעות נחשב את התוחלת של M_s ונקבל :

$$\mathbb{E}_{S\sim\mathcal{D}^m}[\mathrm{err}(\hat{h}_S, \mathcal{D})] = \frac{k-1}{k} \sum_{x\in\mathcal{X}} p_x (1-p_x)^m.$$

החיסרון באלגוריתם זה : אין גודל m של מדגם שיהיה מוצלח עבור כל התפלגות D . ואם $|X| > 2m$ נקבל שהטעות היא לפחות 30%.

<u>סיכום על אלגוריתם ה-*Memorize* :</u>

1. האלגוריתם לא באמת לומד שום דבר הוא רק משנן את התוויות של הדוגמאות שהיו בשלב האימון.
2. אנחנו צריכים לפחות $m = \Omega(|X|)$ כדי לקבל טעות נמוכה.
3. האלגוריתם לא מכליל את מה שראה לדוגמאות שלא ראה (למשל: אם נצפו הרבה נשים שותות הפוך , ומגיעה כעת אישה אז להביא לה הפוך).

<u>הכללה לדוגמאות שלא נצפו :</u>

נגדיר פונקציית מרחק $p : XxX \to R_+$.
בדרך כלל זאת מטריקה שמקיימת :
א. סימטריות $p(x,y) = p(y,x)$.
ב. אי שוויון המשולש $p(x,z) \le p(x,y) + p(y,z)$.
ג. $p(x,y) = 0 \leftrightarrow x = y$.

•<u>איך נבחר את פונקציית המרחק ?</u>
נייצג כל דוגמה ע"י וקטור ב-R^d כך ש-d הוא מספר הפיצ'רים לכל דוגמה.
עבור תכונה כללית (למשל שיער – שחור, בלונד, קצר, צמה..) מומלץ להרחיב את הפיצ'ר לכמה פיצ'רים בינאריים .

$$\rho(x,x') = \|x - x'\| \equiv \sqrt{\sum_{i=1}^{d} (x(i) - x'(i))^2}$$

את המרחק נחשב ע"י נורמת אוקלידיס .

<u>אלגוריתם ה-Nearest neighbor</u>

Nearest Neighbor algorithm

input A training sample S
output A function $\hat{h}_S : \mathcal{X} \to \mathcal{Y}$.
1: Set $\hat{h}_S = f_S^{\mathrm{nn}}$, where f_S^{nn} is defined as:

$$\forall x \in \mathcal{X}, \quad f_S^{\mathrm{nn}}(x) = y_{\mathrm{nn}(x)}.$$

<u>הגדרה :</u> The Bayes-optimal rule הוא ההיפותזה הכי טובה שנוכל לבחור בהינתן התפלגות כלשהי.

$$h_{\mathrm{bayes}} := \text{ any function in } \operatorname*{argmin}_{f\in\mathcal{Y}^{\mathcal{X}}} \mathrm{err}(f, \mathcal{D}).$$

<u>פורמאלי -</u> .

$$\eta_y(x) := \mathbb{P}_{(X,Y)\sim\mathcal{D}}[Y = y \mid X = x] \equiv \mathbb{P}[Y = y \wedge X = x]/\mathbb{P}[X = x].$$

<u>הגדרה:</u>

$$h_{\mathrm{bayes}}(x) \in \operatorname*{argmax}_{y\in\mathcal{Y}} \eta_y(x).$$

<u>טענה:</u> ההיפותזה הכי טובה היא כזו שמקיימת .

- Let $c > 0$. A distribution is "c-nice" with respect to ρ if

$$\forall x, x' \in \mathcal{X}, \quad |\eta(x) - \eta(x')| \leq c \cdot \rho(x, x').$$

הגדרה:

במקרה זה נוכל לטעון כי η היא c-Lipschitz.

הטעות של אלגוריתם ה- *Nearest Neighbor* :

Theorem

If $\mathcal{X} \subseteq [0,1]^d$, $\mathcal{Y} = \{0,1\}$, and η for the distribution $\mathcal{D}$ is c-Lipschitz, then

$$\mathbb{E}_{S\sim\mathcal{D}^m}[\mathrm{err}(f_S^{\mathrm{nn}}, \mathcal{D})] \leq 2\mathrm{err}(h_{\mathrm{bayes}}, \mathcal{D}) + 4c\sqrt{d}m^{-1/(d+1)}.$$

מסקנה : כש-$m \to \infty$ נקבל מ-NN לכל היותר פי 2 מ-Bayes error .
החסם העליון הוא הדוק, יש מקרים כמובן שנקבל פחות מפי 2 טעות . אם d הוא גדול זו יכולה להיות בעיה רצינית .

קללת המימד -The curse of dimensionality

ככל שיש יותר פיצ'רים כך נזדקק ליותר דוגמאות בשלב האימון .

אלגוריתם ה-k-Nearest neighbor

k-**Nearest Neighbors** algorithm

input A training sample S, integer $k \geq 1$.
output A function $\hat{h}_S : \mathcal{X} \to \mathcal{Y}$.
1: Set $\hat{h}_S = f_S^{k\text{-nn}}$, where $f_S^{k\text{-nn}}$ is defined as:

$$\forall x \in \mathcal{X}, \quad f_S^{k\text{-nn}}(x) = \text{the majority label among } y_{\pi_1(x)}, \ldots, y_{\pi_k(x)}.$$

Theorem

Suppose that $k_1, k_2, \ldots$ is a sequence such that $\lim_{m\to\infty} k_m = \infty$, and $\lim_{m\to\infty} k_m/m = 0$. Then

$$\lim_{m\to\infty} \mathbb{E}_{S\sim\mathcal{D}^m}[\mathrm{err}(f_S^{k_m\text{-nn}}, \mathcal{D})] = \mathrm{err}(h_{\mathrm{bayes}}, \mathcal{D}).$$

טענה:

הערה: קללת המימד קיימת גם באלגוריתם הזה!

הערה:
1. חישוב המרחק אם המימד גדול הוא יקר.
2. אחסון כל המידע של הדוגמאות הוא יקר מבחינת זיכרון.

על מנת לפתור את הבעיות האלו הוצעו שתי פתרונות :

- **Principal Components Analysis (PCA)** preserves the general "cloud" shape of the data.
- **Johnson-Lindenstrauss transform (JL)** approximately preserves pairwise distances.

נושא: *Empirical Risk Minimization*

הרעיון הכללי: למצוא היפותזה שעובד טוב על סט האימון S .

פורמאלי : $err(h,S) = \frac{1}{m}\Sigma_{i=1}^{m}\mathbb{I}[h(x_i) \neq y_i]$.

הגדרה: **אלגוריתם *ERM*** הוא אלגוריתם שבוחר היפותזה שממזערת את השגיאה על S .

הגדרה: ***The No Free Lunch theorem*** – נניח כי התוויות הן בינאריות משמע $\mathcal{Y} = \{0,1\}$

Theorem

For any learning algorithm, if $m \leq |\mathcal{X}|/2$, there exists a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$ such that

- *$\mathcal{D}$ has a Bayes-optimal error of zero*
- $\mathbb{E}_{S\sim\mathcal{D}^m}[\mathrm{err}(\hat{h}_S,\mathcal{D})] \geq 1/4$.

בתכלס : אין אלגוריתם טוב שעובד על כל ההסתברויות (ולכן לאנשי *Data Science* עדיין יש עבודה).

הגדרה: *Inductive Bias* הוא אלגוריתם שמכוון ומגביל את אלגוריתם הלמידה באמצעות מידע נוסף שיש לנו על הבעיה שאנחנו מנסים לפתור .

•שיטה פופולארית לאלגוריתם *Inductive Bias* הוא לבחור את ההיפותזה מסט פונקציות מוגבל $H \subseteq \mathcal{Y}^X$.

ERM with a hypothesis class $\mathcal{H}$

Given a training sample $S \sim \mathcal{D}^m$, output $\hat{h}_S$ such that

$$\hat{h}_S \in \underset{h\in\mathcal{H}}{\operatorname{argmin}}\, \mathrm{err}(h,S).$$

הגדרה: *Approximation error=* $err_{app} = \inf_{h\in H} err(h,\mathcal{D})$.

הגדרה: *Estimation error==* $err_{est} = err(\widehat{h_s},S) - \inf_{h\in H} err(h,\mathcal{D})$.

השגיאה הטוטאלית היא $err(\widehat{h_s},\mathcal{D}) = err_{app} + err_{est}$.

The Bias-Complexity tradeoff

ככל שאנחנו מעשירים את קבוצת ההיפותזות H ה- err_{app} קטן אבל ה- err_{est} גדל או צריך כמות דוגמאות גדולה יותר (יש לנו יותר פונקציות בקבוצה H לבחור מהן, יש לנו יותר פונקציות רעות לבחור מהן)
•ככל שאנחנו מגדילים את סט האימון (הדוגמאות) ככה ה- err_{est} קטן יותר (יש יותר דוגמאות ולכן ניתן לבחור פונקציה h יותר "מדויקת") .
ולכן יש לנו איזשהו *tradeoff* בין שתי השגיאות.

הגדרה: ***Overfitting*** מתרחש כאשר השגיאה על סט האימון קטנה ביחס לשגיאה על ההתפלגות (שהיא גדולה). כאשר $err(\widehat{h_S},\mathcal{D}) - err(\widehat{h_S},S)$ הוא גדול.
*זה יכול לקרות אם יש לנו קבוצת היפותזות H עשירה מדיי.

הגדרה: ***Underfitting*** מתרחש כאשר err_{app} הוא גדול . השגיאה על האימון גדולה מדיי.
*זה יכול לקרות אם יש לנו קבוצת היפותזות H שאינה מתאימה לבעיה שלנו.

אז מה האנה מונטנה אמרה ? ***Best of both worlds*** – נבחר קבוצת היפותזות H פשוטה שמתאימה לבעיה שלנו.

בחירת קבוצת ההיפותזות:

•כל קבוצת פונקציות היא חוקית כקבוצת היפותזות H .
•התרחיש הכי טוב- לבחור את קבוצה שמתאימה לבעיה שלנו .

•הבעיה: ברוב המקרים אנחנו לא מספיק מכירים את הבעיה.
• בדרך כלל נשתמש בקבוצת היפותזות כללית .
•בחירות נפוצות- עצי החלטה , *Linear predictors* .

נושא : PAC Learning

השאלה שננסה לענות עליה : בהינתן קבוצת היפותזות H , כמה דוגמאות נצטרך בסט האימון S על מנת להבטיח שאלגוריתם *ERM* על H יחזיר היפותזה עם שגיאה נמוכה על ההתפלגות ?

הגדרה: $\mathcal{D}$ היא *realizable* על H אם קיימת היפותזה $h^* \in H$ כך ש- $err(h^*,\mathcal{D}) = 0$.

•נניח כי : $\mathcal{D}$ היא *realizable* מעל H , וכי לכל x עם הסתברות גדולה מאפס ב-$\mathcal{D}$ להופיע, התווית שלו בהכרח היא $h^*(x)$. (במקרה זה נקבל גם שעבור $S\sim\mathcal{D}^m$, $err(h^*,S) = 0$.
•אם נריץ אלגוריתם *ERM* על $\mathcal{D}$ עם קבוצת ההיפותזות H אז בהכרח נקבל היפותזה $\widehat{h_S}$ כך ש- $err(\widehat{h_S},S) = 0$ אך ייתכן כי $\widehat{h_S} \neq h^*$ (במידה ויש כמה היפותזות שנותנות שגיאה 0, נבחר אחת באקראי ולאו דווקא את h^*).

הגדרה: *Condition parameter* $\delta \in (0,1)$ – החלק מתוך הדוגמאות שנרשה להן להיות "רעות" .

הגדרה: *Error parameter* $\epsilon \in (0,1)$ נחייב שהדוגמאות הטובות יהיו בעלות טעות $err(\widehat{h_S},D) \leq \epsilon$.

למעשה מה שאנחנו מעוניינים לקיים הוא :

$$P_{S\sim D^m}\left[err(\widehat{h_S},D) \leq \epsilon\right] \geq 1-\delta$$

סימון : $(\epsilon.\delta) - sample\ complexity$ קבוצת הדוגמאות בגודל m צריכה לקבל טעות ϵ בהסתברות של $1-\delta$ עבור על התפלגות D .

הגדרה: בהנחה ש-D היא *realizable* על H נגדיר את S להיות ***good sample*** אם לכל $h \in H$ עם $err(h,D) > \epsilon$ נקבל $err(h,S) > 0$.
הגדרה חלופית : S היא *god sample* אם לכל $h \in H$ מתקיים $|err(h,S) - err(h,D)| \leq \frac{\epsilon}{2}$.

אבחנה: אם S היא *good sample* אז אלגוריתם *ERM* מוצא היפותזה h כך ש- $err(h,D) \leq \epsilon$.

Theorem

Let $\epsilon,\delta \in (0,1)$. For any finite hypothesis class $\mathcal{H}$, and any distribution $\mathcal{D}$ over $\mathcal{X}\times\mathcal{Y}$ which is realizable by $\mathcal{H}$, if the training sample size m has

$$m \geq \frac{\log(|\mathcal{H}|)+\log(1/\delta)}{\epsilon},$$

then any ERM algorithm with training sample size m gets an error of at most ϵ, with a probability of at least $1-\delta$ over the random training samples.

הגדרה: *PAC-Probably Approximately Correct*

הערות:
1. לדיוק יותר טוב (אפסילון קטן יותר) נצטרך יותר דוגמאות (קשר ליניארי)
2. לביטחון יותר טוב (דלתא קטן יותר) נצטרך יותר דוגמאות (קשר לוגריתמי) .
3. אם קבוצת ההיפותזות H גדולה יותר, נצטרך יותר דוגמאות בשביל אותם אפסילון ודלתא.

•עבור סט דוגמאות בגודל m נוכל להבטיח שגיאה בגודל $\epsilon \geq \frac{\log\left(|H|+\log\left(\frac{1}{\delta}\right)\right)}{m}$.
עבור קבוצת דוגמאות גדולה וקבוצת היפותזות קטנה נקבל פחות *overfitting* .

Hoeffding's inequality

Let $Z_1,\ldots,Z_m$ be independent random variables over $\{0,1\}$, where for all $i \leq m$, $\mathbb{P}[Z_i = 1] = p$. Then

$$\mathbb{P}\left[\left|\frac{1}{m}\sum_{i=1}^{m} Z_i - p\right| \geq \epsilon\right] \leq 2\exp(-2\epsilon^2 m).$$

The agnostic setting

Make no assumptions on $\mathcal{D}$. Given $\epsilon, \delta \in (0,1)$, require that with probability at least $1-\delta$ over $S \sim \mathcal{D}^m$,

$$\mathrm{err}(\hat{h}_S, \mathcal{D}) \leq \inf_{h \in \mathcal{H}} \mathrm{err}(h, \mathcal{D}) + \epsilon.$$

הגדרה: אפסילון במקרה זה הוא ה-*excess error*.

כעת נבחן כמה דוגמאות נצטרך עבור המקרה האגנוסטי.

Theorem

Let $\epsilon, \delta \in (0,1)$. For any finite hypothesis class $\mathcal{H}$, and any distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, if the training sample size m has

$$m \geq \frac{2\log(|\mathcal{H}|) + 2\log(2/\delta)}{\epsilon^2}$$

then any ERM algorithm with training sample size m gets an excess error of at most ϵ, with a probability of at least $1-\delta$ over the random training samples: $\mathrm{err}(\hat{h}_S, \mathcal{D}) \leq \inf_{h \in \mathcal{H}} \mathrm{err}(h, \mathcal{D}) + \epsilon$.

ההבדל העיקרי בין המקרה האגנוסטי לבין המקרה ה*realizable* היא תלות **ריבועית** באפסילון.

- From $m \geq \frac{2\log(|\mathcal{H}|)+2\log(2/\delta)}{\epsilon^2}$, we get that with a probability $1-\delta$,

$$\mathrm{err}_{\mathrm{est}}(S, \mathcal{H}, \mathcal{D}) \leq \sqrt{\frac{2\log(|\mathcal{H}|) + 2\log(2/\delta)}{m}}.$$

נושא: Infinite hypothesis class

הגדרה: $VC(H) = VC - dimension$ זהו סט הדוגמאות הגדול ביותר שיכול להיות מסווג בכל הקומבינציות האפשריות של התוויות באמצעות קבוצת ההיפותזות *H* .

הערה : $VC(Y^X) = X$ (המחלקה של כל הפונקציות האפשריות)

למעשה *VC(H)* בודק כמה וריאציות אפשריות יש לקבוצת ההיפותזות *H*

- Agnostic case: $m(\epsilon, \delta) = \Theta\left(\frac{\mathrm{VC}(\mathcal{H})+\log(1/\delta)}{\epsilon^2}\right)$.
- Realizable case: $m(\epsilon, \delta) = \tilde{\Theta}\left(\frac{\mathrm{VC}(\mathcal{H})+\log(1/\delta)}{\epsilon}\right)$.

נחזור כעת לאלגוריתם *ERM*. כפי שלמדנו , אלגוריתם זה מחזיר תמיד היפותזה עם שגיאה מינימלית על המדגם. אך אם $|H|$ הוא אקספוננציאלי סיבוכיות החישוב של האלגוריתם עלולה להיות עצומה.

הערות:

1. $2^{VC(H)} \leq |H|$: $for\ all\ finite\ H$ ולכן החסמים עם *VC(H)* הדוקים יותר מאשר $|H|$.
2. קיבלנו כי ה-*sample complexity* של אלגוריתם *PAC* הוא ליניארי ב-*VC(H)* .
3. אם נתבונן ב-*Boolean Conjunctions* נבחין כי $sample\ complexity = O(d)$ אך $|H| = O(3^d)$.

הגדרה: מחלקה היא *learnable* אם יש לה *sample complexity* סופי עבור $(\epsilon, \delta) - PAC\ learning$.

טענה: מחלקה היא *learnable* אם ורק אם יש לה *VC-dimension* סופי .

ERM algorithm for Boolean conjunctions (**realizable setting**)

input A training sample S,
output A function $\hat{h}_S : \mathcal{X} \to \mathcal{Y}$.

```
X_pos ← {x | (x,1) ∈ S}.
Start with conjunction of all literals (h always returns 0)
for x ∈ X_pos do
    for i = 1 to d do
        if x(i) is positive then
            Remove the literal ¬x(i) from conjunction.
        else
            Remove the literal x(i) from conjunction.
        end if
    end for
end for
Return final conjunction
```

•אין אלגוריתם *ERM* יעיל עבור בעיה זו (אלא אם *P=NP*) לכן נחשוב על פתרון אחר.

1. למצוא היפותזה $\widehat{h_S} \in H \to err(\widehat{h_S}, S) > 0$ שגיאה קטנה אך לא 0 (למרות שאנחנו במקרה ה*realizable* ולכן ידוע לנו שקיימת פונקציה *h* כלשהי ב-*H* שכן תספק לנו שגיאה 0).
2. נוכל לנסות להגיע לשגיאה קטנה שווה מאפסילון בהסתברות של $1 - \delta$ במקרה בו כמות הדוגמאות במדגם שלנו מקיים את הנוסחה:

$$m \geq \frac{\log(|\mathcal{H}|)+\log(2/\delta)}{2\epsilon^2}$$

3. כל יוריסטיקה אחרת שתבטיח לנו ש- $err(h, D) \leq err(h, S) + \epsilon$.

Linear predictors: *נושא*

מחלקת היפותזות כללית חשובה היא : *Linear predictors*. מחלקה יסודית וחשובה.

נניח כי $X = R^d$ ונניח שמדובר על תוויות בינאריות $Y = \{0,1\}$.

במקרה זה , $VC - dimension = linear\ in\ d$.

תזכורת מאלגברה:

הגדרה: מכפלה פנימית. לכל וקטור $x, z \in R^d, < x, z > := \Sigma_{i=1}^{d} x(i) * z(i)$.

נומרה: $||x|| = \sqrt{< x, x >}$.

לשם הנוחות נקרא לתוויות $Y = \{-1,1\}$.

הגדרה: עבור $w \in R^d, b \in R$ פרדיקטור ליניארי מוגדר להיות
$\forall x \in R^d\ , h_{w,b}(x) = sign(< w, x > +b)$.

כך ש- *b* מוגדר להיות ה-***bias*** של הפרדיקטור.

הגדרה: מחלקת ההיפותזות של הפרדיקטורים הלינארים במימד *d* הוא

$$H_{LB}^d := \{h_{w,b} | w \in R^d, b \in R\}$$

•ה-*b* לא נדרש. נניח שהבעיה היא עם $X = R^d$, לכל דוגמה $x \in R^d$ נגדיר $x' \in R^{d+1}$.
לכל פרדיקטור ליניארי עם *bias* ב-R^d נגדיר פרדיקטור ליניארי בלי *bias* ב-R^{d+1}:
$w' := (w(1) \ldots . w(d), b)$

•ה-*b* מעצבן, ומפריע לחישובים ולכן לרוב נעדיף לעבוד בצורה השקולה בהוספת המימד וללא ה-*b* עצמו.

הערה: פרדיקטורים ליניאריים ללא *bias* נקראים ***homogeneous*** . ופרדיקוטורים ליניאריים עם *bias* נקראים גם ***halfspaces*** .

מימד ה-*VC* עבור *linear predictors* :

טענה: ב-R^d קיימת קבוצה שניתנת לניתוץ בגודל *d* .

מסקנה : $VC-dimension(H_L^d) \geq d$.

טענה: לא קיימת קבוצה ניתנת לניתוץ ב-R^d בגודל *d+1* .

מסקנה: $VC-dimension(H_L^d) = d$.

טענה: עבור קבוצת הפרדיקטורים הליניאריים עם *bias* , $VC-dimension(H_{LB}^d) = d+1$.

***Sample complexity* עבור *linear predictors* :**

- The VC dimension of homogeneous linear predictors is d.
- Realizable sample complexity:

$$m(\epsilon,\delta) = \tilde{\Theta}\left(\frac{\mathrm{VC}(\mathcal{H}_L^d) + \log(1/\delta)}{\epsilon}\right) = \Theta\left(\frac{d + \log(1/\delta)}{\epsilon}\right)$$

- Agnostic sample complexity:

$$m(\epsilon,\delta) = \Theta\left(\frac{\mathrm{VC}(\mathcal{H}_L^d) + \log(1/\delta)}{\epsilon^2}\right) = \Theta\left(\frac{d + \log(1/\delta)}{\epsilon^2}\right).$$

- In both cases, the sample complexity is **linear in** d.
- $\Rightarrow$ a sample size linear in d suffices to avoid overfitting.

מסקנה: אם יש פרדיקטור ליניארי עם שגיאה נמוכה עבור התפלגות *D* וכמות הדוגמאות מספיק גדולה (ליניארי ב-*d*) אזי בהסתברות גבוהה אלגוריתם *ERM* עבור מחלקת ההיפותזות H_L^d יחזיר היפותזה עם שגיאה נמוכה.

מימוש אלגוריתם *ERM* באמצעות *linear predictors*:

עבור פרדיקטורים ליניאריים שהם *homogeneous* אלגוריתם *ERM* יהיה :

$$\mathrm{Minimize}_{w\in\mathbb{R}^d}\,\mathrm{err}(h_w, S), \text{ where}$$
$$\mathrm{err}(h_w, S) := \frac{1}{m}\left|\{i \mid \mathrm{sign}(\langle x_i, w\rangle) \neq y_i\}\right|.$$

•זו בעיה שהיא *NP*-קשה ☹

ננסה לפתור את הבעיה במקרה ה-*realizable* (כי במקרה זה אכן קיים פתרון כללי לבעיה הזו)

הגדרה: *Linear program* היא בעיה מהצורה הבאה:

$$\begin{aligned} &\text{maximize}_{w\in\mathbb{R}^d} && \langle u, w\rangle \\ &\text{subject to} && Aw \geq v. \end{aligned}$$

כאשר $w, u \in R^d, v \in R^m, A \in R^{mxd}$ והם מגדירים את הבעיה הליניארית.

איך נמיר את הבעיה שלנו לתכנון ליניארי?

נכתוב את הבעיה בצורה הזו : $find\ w \in R^d: \forall i \leq m, y_i < x_i, w >> 0$

בעיה: אנחנו צריכים גדול ממש ולא גדול שווה, זה יכול לגרום לווקטור האפס לחזור כפתרון.

נשנה את הבעיה לצורה הזו : $find\ w \in R^d: \forall i \leq m, y_i < x_i, w > \geq 1$.

טענה: שינוי הבעיה מתאים לפתור את הבעיה שלנו.

1. כל פתרון שפותר גם את הבעיה החדשה פותר גם את הבעיה הישנה.
2. אם יש פתרון לבעיה הישנה אז יהיה פתרון גם לבעיה החדשה .

מסקנות על תכנון ליניארי :

1. תכנון ליניארי פשוט ליישום אבל יכול להיות איטי.
2. כל הדוגמאות חייבות להיות מאוחסנות בזיכרון.
3. אם יש אפילו דוגמה אחת בעייתית האלגוריתם יכול להיכשל לחלוטין (במקום למשל לספק פתרון עם שגיאה קטנה)

נושא: *The perceptron*

•הגרסה הזו נקראת *Batch perceptron*.

פואנטה : למצוא פרדיקטור ליניארי עם שגיאה 0 על מדגם האימון.
איך תכלס? נתחיל מפרדיקט דיפולטיבי , וכל עוד יש דוגמה במדגם האימון שהפרדיקט טועה עליה נבצע סיבוב לפרדיקט "בכיוון הנכון" . (שימו לב-אף אחד לא מבטיח שלאחר התיקון הפרדיקט יהיה נכון על הדוגמה אך הוא יהיה פחות טועה).

האלגוריתם:

Batch Perceptron

input A **separable** training sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$
output $w \in \mathbb{R}^d$ such that $\forall i \leq m, h_w(x_i) = y_i$.

```
t ← 1, w^(1) ← (0, ..., 0)
while ∃i s.t. y_i⟨w^(t), x_i⟩ ≤ 0 do
    w^(t+1) ← w^(t) + y_i x_i
    t ← t + 1
end while
Return w^(t).
```

למה זה הגיוני ? בכל עדכון הפרדיקט זז $y_i < w^{(t)}, x_i >$ קרוב יותר להיות חיובי.

•אינטואיטיבית נבחין כי ההפרדה הזו ע"י פרדיקטור ליניארי היא קלה יותר לביצוע כאשר הנקודות החיוביות והשליליות (עם תווית מנוגדת) רחוקות יותר אחת מהשנייה .

Claim

$\frac{|\langle w,x\rangle|}{||w||}$ is the **distance** between $x \in \mathbb{R}^d$ and the separator defined by w.

The separation margin

נגדיר $s.t : \left|\left|w\right|\right| = \sqrt{< \overline{w}, \overline{w} >} = 1$ $\overline{w} := \frac{w}{||w||}$.

ה-*hyperplane* מוגדר להיות $H = \{v| < v, w >= 0\} = \{v| < v, \overline{w} >= 0\}$.

המרחק בין ה*hyperplane* לבין *x* מוגדר להיות: $\Delta := \min_{v \in H} ||x - v||$.

טענה: $\Delta = | < \overline{w}, x > | = \frac{|<w,x>|}{||w||}$.

הגדרה: $R := \max_i ||x_i||$ (*R* מהמילה רדיוס, מדובר ברדיוס הכדור שמקיף את כל הנקודות במדגם האימון).

הגדרה: המרחק המנורמל המינימאלי עבור $x_i \in S$ מ-w נקרא ה-*margin* של w .

$$\gamma(w) := \frac{1}{R}\min_{i\le m}\frac{|<w,x>|}{||w||}$$

Theorem

Assume that $S = ((x_1, y_1), \dots, (x_m, y_m))$ is separable. Then

1. *When the Perceptron stops and returns $w^{(t)}$, $\forall i \le m, y_i\langle w^{(t)}, x_i\rangle > 0$.*
2. *Define: $\gamma_S := \max\{\gamma(w) \mid w \in \mathbb{R}^d, w$ separates $S\}$.*
   Then, the Perceptron performs at most $1/\gamma_S^2$ updates.

מסקנות מהוכחת המשפט:

- Let w^* such that $\min_{i\in[m]} y_i\langle w, x_i\rangle = 1$ and $\gamma(w^*) = \gamma_S$.
  - Such a w^* must exist: can always rescale by $\min_{i\in[m]} y_i\langle w, x_i\rangle$ without changing the margin.
- Define: $B := \|w^*\|$. So $\gamma_S = 1/(RB)$.

ונסמן T להיות כמות האיטרציות שהאלגוריתם מבצע.

We showed

- The Perceptron makes at most $(RB)^2$ updates.
- $RB = 1/\gamma_S$.
- When it stops, the separator it returns separates the examples in S.
- So, number of updates until finding a separator is at most $1/\gamma_S^2$.
- Returned separator not necessarily with best margin!

טענה: ה-*sample complexity* של אלגוריתם ה-*Perception* הוא $O(\frac{1}{\gamma_d^2})$.

לסיכום:

1. מספר האיטרציות של העדכון תלוי ב-*sample margin* .
2. מעבדים דוגמה אחת בכל שלב, אין צורך לאחסן את כל הדוגמאות בזיכרון במקביל.
3. אם ה-*margin* מאוד קטנים אזי הסיבוכיות של האלגוריתם היא $\Omega(2^d)$.
4. אם הדוגמאות במדגם האימון לא ניתנים להפרדה אזי האלגוריתם *LP* עלול להיכשל לחלוטין , לעומת זאת אלגוריתם ה*Perception* עדיין ירוץ אבל לא בהכרח יסיים לרוץ בכוחות עצמו. אין הבטחות על אלגוריתם ה-*Perception* במקרה זה אבל עדיין אפשר לנסות .
5. בפועל , אלגוריתם ה*Perception* יכול לרוץ מהר יותר מאלגוריתם *LP* .
6. *Sample complexity* : האלגוריתם הוא אלגוריתם *ERM* לכל דבר (במקרה ה-*realizable*) ובמקרה זה ה-*sample complexity* יהיה $\theta(d)$ אבל אלגוריתם ה-*Perception* יכול לרוץ מהר יותר אם ה-*distribution margin* הוא גדול.

נושא : *Hard SVM*

Large-margin algorithm (for the separable case)

Select the separating linear predictor with maximal margin on sample.

Sample margin

- $R := \max_i \|x_i\|$,
- $\gamma(w, S) := \frac{1}{R}\min_{i\le m}\frac{|\langle w, x_i\rangle|}{\|w\|}$,
- $\gamma_S := \max\{\gamma(w, S) \mid w$ separates $S\}$.

Distribution margin

- $R_{\mathcal{D}} := \max_{x \in \mathrm{supp}(\mathcal{D}_X)} \|x\|$
- $\gamma(w, \mathcal{D}) := \frac{1}{R_{\mathcal{D}}} \inf_{x \in \mathrm{supp}(\mathcal{D}_X)} \frac{|\langle w, x \rangle|}{\|w\|}$.
- $\gamma_{\mathcal{D}} := \sup\{\gamma(w, \mathcal{D}) \mid \mathrm{err}(h_w, \mathcal{D}) = 0\}$.

אלגוריתם *Hard SVM*

•אלגוריתם זה מקבל *sample* שניתן להפרדה (כלומר, הבעיה הינה *realizable*). ומוצא את המפריד המקסימאלי באמצעות *Quadratic Program*.

Hard-SVM

input A separable training sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$
output $\hat{w} \in \mathbb{R}^d$ such that $\forall i \leq m, h_w(x_i) = y_i$.
1: Find $\hat{w}$ that solves the following problem:

$$\text{Minimize}_{w \in \mathbb{R}^d} \|w\|^2$$
$$\text{s.t. } \forall i, y_i \langle w, x_i \rangle \geq 1.$$

2: Return $\hat{w}$.

טענה: אלגוריתם *Hard SVM* מחזיר $\hat{w}$ אשר ממקסם את שולי המפריד .

נושא: *Quadratic Program*

Quadratic Program

A QP is a problem of the following form:

$$\text{minimize}_{w \in \mathbb{R}^d} \quad \frac{1}{2} w^T \cdot H \cdot w + \langle u, w \rangle$$
$$\text{subject to} \quad Aw \geq v.$$

כך ש- $w \in R^d$ הווקטור שאנחנו מעוניינים למצוא , $u \in R^d, H \in R^{dxd}, v \in R^m, A \in R^{mxd}$.
*שימו לב: H חייבת להיות מטריצה *positive semi-definite* .
•אלגוריתם *QP* כול להיפתר בצורה יעילה.

Quadratic program	Hard-SVM minimization problem
$\text{minimize}_{w \in \mathbb{R}^d} \quad \frac{1}{2} w^T \cdot H \cdot w + \langle u, w \rangle$ $\text{subject to} \quad Aw \geq v.$	Minimize $\|w\|^2$ s.t. $\forall i, y_i \langle w, x_i \rangle \geq 1.$

במקרה זה $u = (0, \ldots, 0)$, $H = 2 * I_d$ ומטריצה A תהיה כך שהשורה ה-i במטריצה היא
$(y_i * x_i(1), \ldots, y_i * x_i(d))$

הגדרה: *Support vector* הם קבוצה של נקודות שנמצאות על המפריד המקסימאלי.
מוגדרים להיות כל הווקטורים $x_i \in S$ כך ש- $|\langle \hat{w}, x_i \rangle| = 1$.

תיאוריה: $\hat{w}$ הוא קומבינציה ליניארית של *support vectors* :
$$\exists \alpha_1, \ldots, \alpha_{|I|} \in R, \hat{w} = \Sigma_{i=1}^{|I|} \alpha_i x_i$$

כל תת-קבוצה בלתי תלויה של *support vectors* בגודל d מספיקה על מנת לתאר את $\hat{w}$.

Sample Complexity:

•*Hard SVM* משפר את הסיבוכיות לעומת *ERM* אם $\frac{1}{\gamma_D^2} \ll d$.

•מכיוון שאלגוריתם זה הוא *ERM* למעשה סיבוכיות החישוב שלו היא $O(\min\left(d, \frac{1}{\gamma_D^2}\right))$.

אלגוריתם *Soft SVM*

במקרה הכללי, שהבעיה אינה *realizable* לא יהיה מפריד מושלם (שיחזיר שגיאה 0) אך היינו רוצים מפריד שיחזיר שגיאה נמוכה על המדגם (לאו דווקא המינימאלית). נראה מפריד עם שגיאה נמוכה ו-*margin* גדול.
במקרה זה הבעיה היא *NP*-קשה ☹
מעוניינים למזער את השגיאה על המדגם אך לא יודעים איך לעשות את זה. נגדיר פונקציית מטרה שאותה אנחנו כן נצליח למזער.

הגדרה: פונקציית *Loss* . פונקציה בעלת קלט $w \in R^d$ ודוגמה (x,y) ומחזירה מספר אי-שלילי שמודד את ההפסד של *w* על (x,y).

הגדרה: *sample loss* - $\ell(w,S) := \frac{1}{m}\Sigma_{i\in[m]}\ell(w,(x,y))$.

הגדרה: *distribution loss* - $\ell(x,D) := \mathbb{E}_{(x,y)\sim D}\ell(w,(x,y))$.

הגדרה: *zero-one loss* מוגדר להיות $\ell_{0-1}(w,(x,y)) := \mathbb{I}[h_w(x) \neq y]$.

$$\ell(w,S) = \mathrm{err}(h_w,S) \quad \text{and} \quad \ell_{0-1}(w,\mathcal{D}) = \mathrm{err}(h_w,\mathcal{D}).$$

אלגוריתם *ERM* למעשה ממזער את *zero-one loss* . את הפונקציה הזו אנחנו לא יודעים למזער ולכן נציע פונקציית הפסד חלופית שאותה כן נצליח למזער בצורה יעילה, ושהפונקציה תהיה קשורה לפונקציית *zero-one* ככל שאפשר.

האלגוריתם החדש – ימזער את הפונקציה $\ell^h(w,S)$ ונקבל פרדיקטור ליניארי $\hat{w} \in argmin_{w\in R^d}\ell^h(w,S)$.
בניגוד לאלגוריתם *ERM* אנחנו לא בהכרח נתקרב לפתרון שיביא לנו את $\inf_{w\in R^d} err(h_w,\mathcal{D})$.

הרעיון של *Soft margin*:

לעומת *hard SVM* שבו נבחר מפריד אם ורק אם הוא יהיה בעל שגיאה 0 על המדגם, ויתבסס על המרחק של הנקודות הכי קרובות למפריד. במקרה זה, נאפשר למפריד לבצע שגיאות על המדגם אך על השגיאה אנו "נשלם". כלומר, אם יהיה לנו טעויות גדולות- ככל שהן יותר רחוקות מהמפריד- נשלם יותר. נשלם גם עבור סיווג נכון של תוויות שנמצאות בתוך ההפרדה .
הרוחב של ההפרדה יהיה תלוי בנורמה של *w* . (ביחס הפוך – ככל שהנורמה קטנה יותר כך ההפרדה גדול יותר).

הגדרה: $hinge\ loss : \ell^h(w,(x,y)) = \max\{0,1-y<w,x>\}$.

•קל למזער את הפונקציה ההפסד הזו.
• תכונות טובות מבחינת *sample complexity* .
• קיים יחס בין פונקציית ההפסד ℓ^h לבין השגיאה $error_{0-1}$:

$$\forall w,x,y\ \ell_{0-1}(w,(x,y)) = \mathbb{I}[h_w(x) \neq y] \leq \ell^h(w,(x,y))$$

מסקנה: *hinge loss* חוסם מלמעלה את *zero-one loss* .

•עבור אותו מפריד , עם נורמות שונות נקבל *hinge loss* שונה .

האזור של הדוגמאות עבורן נשלם יהיה : $\psi := \{x||<w,x>| \leq 1\}$.

במקרה זה הרוחב של ה-*soft margin* יהיה: $\max_{x\in\psi}\frac{|<w,x>|}{||w||} = \frac{1}{||w||}$.

הגדרה: *sample hinge loss* מוגדר להיות

$$\ell^h(w,S) := \frac{1}{m}\sum_{i=1}^{m}\ell^h(w,(x_i,y_i)).$$

•נבחן מקרה פשוט: S הוא ניתן להפרדה- עבור $\alpha > 0$ גדול מספיק נקבל כי $\ell^h(\alpha w, S) = 0$.
במקרה זה, נבחר מפריד כלשהו עם נורמה גדולה מספיק שממזערת את *hinge loss* .
לכן למזער רק את *hinge loss* לא יבחר מפריד בעל *margin* גדול.
במקרה זה נקבל *ERM* שרירותי והסיבוכיות תהיה $O(d)$.
בעיה זו קיימת גם במקרה הלא ניתן להפרדה (פשוט קשה יותר להראות את זה)
•איך נפתור את הבעיה ? ננסה למצוא דרך למזער את הנורמה של w וגם את ה-*hinge loss* .

<u>האלגוריתם:</u>

<u>מטרת האלגוריתם:</u> למצוא w עם עונש קטן. בניגוד ל*hard SVM* במקרה זה מדגם האימון S אינו חייב להיות ניתן להפרדה .

<u>הגדרה:</u> העונש עבור w יוגדר להיות $P(w) = \lambda||w||^2 + \ell^h(w, S)$
עבור $\lambda > 0$ פרמטר שיתקבל כקלט לאלגוריתם (ערכו יסמל למעשה את החשיבות שאנו נותנים לנורמה של w ביחס ל*hinge loss*).

Soft-SVM

input A training sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, parameter $\lambda > 0$.
output $w \in \mathbb{R}^d$
1: Find w that solves the following problem:

$$\text{Minimize}_{w \in \mathbb{R}^d, \xi_i \in \mathbb{R}^m} \quad \lambda\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m} \xi_i$$

$$\text{s.t. } \forall i, y_i\langle w, x_i\rangle \geq 1 - \xi_i, \text{ and } \xi_i \geq 0.$$

2: Return w.

<u>טענה:</u> אלגוריתם *Soft-SVM* מחזיר w אשר ממזער את $P(w)$.

על מנת לייצג את הבעיה שלנו בתוך *Quadratic program* נבין מי הם z, A, v, u, H.

$$z = (w(0), \ldots w(d), \mu_1, .. \mu_m)$$
$$u = \left(0, \ldots 0, \frac{1}{m}, \frac{1}{m}, \ldots, \frac{1}{m}\right) \ \#d \ times \ zero \ and \ m \ times \frac{1}{m}$$

$$v = (0, \ldots, 0, 1, \ldots, 1) \#m \ times \ zero \ and \ m \ times \ one$$

$$H \in R^{(d+m)x(d+m)}$$

כך שהבלוק השמאלי עליון בגודל *dxd* יהיה מטריצת $2\lambda I_d$.

$$A \in R^{(2m)x(d+m)}$$

כך שהבלוק השמאלי עליון בגודל *mxd* הוא אפסים, הבלוק הימני עליון וימני תחתון בגודל *mxm* הוא מטריצת יחידה והבלוק השמאלי תחתון בגודל *mxd* כך שהשורה ה-*i* במטריצה היא $(y_i x_i(1), \ldots, y_i x_i(d)$.

<u>Sample Complexity of Soft SVM</u>

•מה שמעניין אותנו זה $err(h_w, \mathcal{D})$ אבל אנחנו ממזערים את ה-*hinge loss* על המדגם.
ראינו ש- $err(h_w, \mathcal{D}) \leq \ell^h(w, \mathcal{D})$.

<u>תזכורת:</u> $R_D = maximum \ norm \ of \ x \in \sup(\mathcal{D})$. (גורם נרמול)
<u>תיאוריה:</u> לפרדיקטור שממזער את *soft SVM* על המדגם יש *hinge loss* נמוך על ההתפלגות.

Theorem

Suppose that $\hat{w}_S$ is the output of the soft-SVM algorithm on the input sample S. Then,

$$\mathbb{E}_{S\sim\mathcal{D}^m}[\ell^h(\hat{w}_S, \mathcal{D})] \leq \min_{u\in\mathbb{R}^d}\left(\ell^h(u, \mathcal{D}) + \lambda\|u\|^2\right) + \frac{2R_{\mathcal{D}}^2}{\lambda m}.$$

•הפונקציה h_w היא הפונקציה שמבצעת את ההכפלה בין ווקטור *w* לבין *x* על מנת לקבל מספר ממשי כלשהו ולפיו לקבוע את התווית של *x* .

<u>שאלה: מהו הערך הכי טוב עבור λ ?</u>

על מנת למצוא את החסם העליון הקטן ביותר , נמצא $\lambda^* = argmin_{\lambda>0}\lambda B^2 + \frac{2R_D^2}{\lambda m}$.

נקבל: $\lambda^* = \sqrt{\frac{2R_D^2}{B^2 m}}$.

Guarantee with optimal λ

$$\mathbb{E}_{S\sim\mathcal{D}^m}[err(h_{\hat{w}_S},\mathcal{D})] \le \min_{u:\|u\|\le B} \ell^h(u,\mathcal{D}) + \sqrt{\frac{8R_{\mathcal{D}}^2 B^2}{m}}.$$

<u>מסקנה:</u> *sample complexity* היא $O(R_D^2 B^2)$.
נבחין כי אין תלות במימד *d* .

כעת נותרת השאלה איזה *B* כדאי לבחור? מדובר פה בסוג של מחלקת היפותזות שמוגדרת להיות :
כל הפרדיקטורים שהנורמה שלהם קטנה או שווה מ-*B*.

<u>מדובר ב*tradeoff* :</u>
(המחוברים הכוונה היא למחוברים בחסם העליון על השגיאה הצפויה)
1. *B* קטן -המחובר הראשון עלול להיות גדול, נקבל err_{app} גבוה.
2. *B* גבוה- המחובר השני עלול להיות גדול, נקבל err_{ext} גבוה.

<u>לסיכום:</u>

Algorithm	Assumption	Objective	Sample Complexity	Efficient?
Full ERM	None	ERM	$\Theta(d)$	No, NP-hard
LP	Realizable	ERM	$\Theta(d)$	Yes
Perceptron	Realizable	ERM	$O(1/\gamma_{\mathcal{D}}^2)$	Sometimes
Hard-SVM	Realizable	large margin	$O(1/\gamma_{\mathcal{D}}^2)$	Yes
Soft-SVM	None	small hinge-loss	$O(B^2) \equiv 1/(\text{soft margin})^2$	Yes

*ב-*sample complexity* של אלגוריתם *Perceptron* ו-*Hard SVM* מדובר למעשה ב- $\min(d, \frac{1}{\gamma_{\mathcal{D}}^2})$.

<u>נושא: *Validation and cross-Validation*</u>

<u>פרמטרים של אלגוריתם למידה:</u>

להרבה אלגוריתמי למידה יש פרמטרים בקלט כמו:
1. *K* עבור אלגוריתם *k-NN*.
2. λ עבור אלגוריתם *soft-SVM* .

פרמטרים אלו יוצרים *trade-off* בין גורמים באלגוריתם, ולמעשה הפרמטר האופטימלי תלוי על ההתפלגות הלא ידועה $\mathcal{D}$.

<u>*Validation sample*</u>

• נעריך את הערך של הפרמטר בקלט ע"י *validation sample* .

<u>הגדרה:</u> *validation sample* מסומן כ- $V = ((x_1', y_1') \dots . (x_n', y_n'))$ מדגם אימון מתויג שלא משומש עבור שלב האימון באלגוריתם. $V\sim\mathcal{D}^n$.

•נשתמש בשגיאה שנקבל על ה-*validation sample* על מנת לקבל הערכה ל-$err(\widehat{h_s},\mathcal{D})$.

אלגוריתם:

Choosing a parameter using a validation set

input A training sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$,
A validation set $V = \{(x'_1, y'_1), \ldots, (x'_n, y'_n)\}$,
A learning algorithm $\mathcal{A}$ with a parameter α,
Ψ = A **finite set** of possible values for α.
output $\hat{h}_S : \mathcal{X} \to \mathcal{Y}$,
1: **for** $\alpha \in \Psi$, **do**
2: $\hat{h}_\alpha \leftarrow \mathcal{A}(S, \alpha)$
3: **end for**
4: Select $\hat{\alpha} \in \operatorname{argmin}_{\alpha \in \Psi} \operatorname{err}(\hat{h}_\alpha, V)$.
5: Output $\hat{h}_S = \hat{h}_{\hat{\alpha}}$.

•על פי האלגוריתם, אנו משתמשים ב- $err(\widehat{h_\alpha}, V)$ על מנת להעריך $err(\widehat{h_\alpha}, \mathcal{D})$.

כמה טובה ההערכה הזו ?

נגדיר $Z_i^\alpha = \mathbb{I}[\widehat{h_\alpha}(x'_i) \neq y'_i]$ משתנה מקרי אינדיקטור שתלוי ב-V.

$\mathbb{P}[Z_i^\alpha = 1] = err(\widehat{h_\alpha}, \mathcal{D})$ ומתקיים ש- $Z_1^\alpha, \ldots, Z_n^\alpha$ הם ב"ת. ולכן נקבל :

$$err(\widehat{h_\alpha}, V) = \frac{1}{n} \Sigma_{i=1}^n Z_i^\alpha$$

מסקנה: לכל $h \in \mathcal{H}$ שלא תלויה ב-V נקבל כי :

$$\mathbb{P}[|err(h, V) - err(h, \mathcal{D})| \geq \epsilon] \leq 2e^{-2\epsilon^2 n}$$

•נוכיח כי לכל $\alpha \in \psi$ השגיאה המשוערת היא בסדר. (ולכן בפרט השגיאה המשוערת עבור ה- α שהאלגוריתם בחר היא בסדר).

הגדרה: אירוע $E(\alpha)$ לכל $\alpha \in \psi$: $E(\alpha)\ \ is\ true\ iff\ \left|err(\widehat{h_\alpha}, V) - err(\widehat{h_\alpha}, \mathcal{D})\right| \geq \epsilon$.

$$\mathbb{P}[E(\alpha)\ holds] \leq 2\ |\ \psi|e^\wedge(-2\epsilon^2 n)$$

מסקנה: $\forall \delta \in (0,1)$: $n \geq \frac{\log\left(\frac{2|\psi|}{\delta}\right)}{2\epsilon^2}$ אזי $\mathbb{P}[\forall \alpha \in \psi\ , |err(h, V) - err(h, \mathcal{D})| \leq \epsilon] \geq 1 - \delta$.

במקרה זה, השגיאה הכי טובה היא לכל היותר $2\epsilon = \frac{\sqrt{2\log\left(\frac{2|\psi|}{\delta}\right)}}{n}$.

•גודל סט ה-*validation* שנצטרך תלויה בצורה לוגריתמית במספר הפרמטרים $|\psi|$.

•לא נוכל לבדוק כמות אינסופית של פרמטרים.

The holdout set

•בפועל, יש לנו קבוצה מתויגת אחת בלבד. ואנחנו צריכים להשתמש בחלק ממנו ל-*validation*.

•מחשבים עבור הפרמטר שמצאנו את הפרדיקטור לפי כל המדגם.

•מחלקים את הסט המתויג לסט אימון ולסט *validation* לפי השיקולים שלנו.

•אין כעת הבטחות, אנחנו יודעים שהשגיאה על V קרובה לשגיאה על V אך לא ניתן לדעת אם $\widehat{h_S} = \mathcal{A}(S, \hat{\alpha})$ דומה ל- $\widehat{h_{\hat{\alpha}}} = \mathcal{A}(S', \hat{\alpha})$.

Learning with parameters using a holdout set

input A training sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$,
A learning algorithm $\mathcal{A}$ with a parameter α,
Ψ = A finite set of possible values for α.
output $\hat{h}_S : \mathcal{X} \to \mathcal{Y}$,

```
Split the training sample into S' and V
for α ∈ Ψ, do
    ĥ_α ← A(S', α)
end for
Select α̂ ∈ argmin_{α∈Ψ} err(ĥ_α, V).
Run ĥ_S ← A(S, α̂).
Output ĥ_S.
```

K-Fold Cross-Validation

•האם ניתן לבחור את α בצורה מדויקת יותר?
כן! נשתמש בכמה *holdout sets* ונבחר את הממוצע של השגיאה המשוערת.
ה- *holdout sets* נוצרים ע"י חלוקת סט האימון *S* מספר פעמים.

Learning with parameters using k-fold cross-validation

input A training sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$,
A learning algorithm $\mathcal{A}$ with a parameter α,
Ψ = A finite set of possible values for α,
k = number of folds.
output $\hat{h}_S : \mathcal{X} \to \mathcal{Y}$.

```
Split the training sample into k equal parts, S_1, ..., S_k.
for α ∈ Ψ, do
    for i ∈ {1, ..., k} do
        V ← S_i, S' ← S \ S_i
        ĥ_i ← A(S', α)
        ε_i ← err(ĥ_i, V).
    end for
    ε_α = (1/k) Σ_{i=1}^{k} ε_i
end for
Select α̂ ∈ argmin_{α∈Ψ} ε_α.
Run ĥ_S ← A(S, α̂).
Output ĥ_S.
```

•לאלגוריתם זה אין הבטחות. אך ברוב המקרים זה כן יעבוד טוב.

•ניתן להראות שעבור *k* גדול יותר, נקבל שגיאה משוערת מדויקת יותר.

•ערכים סטנדרטים ל-*k* הם : 5 או 10. מקרה קיצוני הוא לבחור $k = m$ – וזהו למעשה המקרה הכי מדויק אך יקר מאוד לחישוב.

נושא : *Kernels*

מוטיבציה: חישובים מסובכים ניתנים לפתרון ע"י *kernels* -
• *kernels* הן פונקציות שמאפשרות חישובים מהירים במימד גבוה.

Non-linear Classifiers

•ייתכן שמפריד ליניארי לא יעבוד , האם נוכל להפוך את הבעיה לפרידה ליניארית בכל זאת?
כן! אם נשנה את הייצוג את הבעיה כך : $\forall x \in R \to use\ \psi(x) = (x, x^2) \in R^2$.
עבור הייצוג החדש, יש מפריד ליניארי.

הגדרה: *Feature maps* רעיון כללי עבור שינוי הייצוג של בעיית הלמידה.
תהי $\mathcal{X}$ התחום של הדוגמאות ויהי $\mathcal{F} = R^n$ להיות מימד הפיצ'רים החדש.
n^* הוא בדרך כלל גדול ולעיתים אינסופי.
feature map מוגדרת להיות פונקציה $\psi: \mathcal{X} \to \mathcal{F}$.

בהינתן סט דוגמאות S, התמונה של הסט הזה במימד פיצ'רים החדש הוא :
$\hat{S} = ((\psi(x_1), y_1), \dots, (\psi(x_n), y_n))$

ה-*feature map* יוצר התפלגות חדשה, נסמנה $\mathcal{D}^{\psi}$.

<u>טענה:</u> $\forall h\,, err(h, \mathcal{D}^{\psi}) = err(h \cdot \psi, \mathcal{D})$.

•<u>נגדיר את קבוצת ההיפותזות</u>
$\mathcal{H}_{\psi} = \{h_w \cdot \psi \,|w \in \mathcal{F}\} = \{x \to sign(< w, \psi(x) >)|w \in R^n\}$ כך ש- $\mathcal{H}_{\psi} \subseteq \mathcal{Y}^{\mathcal{X}}$.

•מצב טוב מבחינתנו יהיה : קיים מפריד ליניארי עם *margin* גדול במרחב הפיצ'רים החדש.

•אבחנה: ה-*feature map* נבחר לפני שאנחנו מנתחים את סט האימון. (אחרת יכול להיווצר *overfitting*)

<u>*Polynomial feature maps*</u>

ראשית נבחן את זה כאשר $\mathcal{X} = \mathbb{R}$.
פולינום מדרגה k עבור $x \in \mathbb{R}$ היא פונקציה $f: \mathbb{R} \to \mathbb{R}$ מהצורה:
$$f(x) \coloneqq a_0 + a_1 x + \dots + a_k x^k = \Sigma_{i=0}^{k} a_i x^i \;\; for\ some\ a_i \in \mathbb{R}$$

יהי $\mathcal{F} = \mathbb{R}^{k+1}$ אזי נגדיר $\psi: \mathbb{R} \to \mathbb{R}^{k+1}$ כך : $\psi(x) = (1, x, \dots., x^k)$.

כל מפריד ליניארי $w \in \mathbb{R}^{k+1}$ כעת מתאר מפריד לפי הפולינום מדרגה k :
$$h_w(\psi(x)) = sign(\Sigma_{i=1}^{(k+1)} w_i x^{i-1}$$

•עבור $\mathcal{X} = \mathbb{R}^d$: נפריד באמצעות *multivariate polynomials*.

•*multivariate polynomial* מדרגה k הוא **סכום** ה-*monomials* מדרגה k .

•*monomials* : כל הכפלה של המשתנה מהקלט כך שסכום החזקות של איברי המכפלה הוא לכל היותר k .
<u>לדוגמה :</u> עבור הקלט $x \in R^d$ *monomial* מדרגה 3 אפשרית היא : $7x(1)^2x(4)$ כך ש-7 הוא מקדם סקלארי ו$(1) = x(1), (4) = x(4)$.

<u>הגדרה:</u> $I_d^k = \{t \in \mathbb{N}^d \,|\Sigma_{j=1}^{d} t(j) \le k\}$
ועכשיו בגרסה המובנת – כל הסדרות באורך d המכילות איברים בין 0 ל-k כך שסכום איברי הסדרה הוא לכל היותר k .

ואם נקשר את ההגדרה הזו ל-*monomial* למעשה כך מונומיאל הוא :
$$c * \prod_{i=1}^{d} x(i)^{t(i)}$$

במקרה זה- c הוא המקדם הסקלארי , $t(i)$ זה הערך של הסדרה t באינדקס ה-i , ו- $x(i)$ זו הקורדינטה ה-i בווקטור x .

הגדרה: קבוצת כל ה-*multivariate polynomial* מדרגה k הם :

$$\{x \to \Sigma_{t \in I_d^k} a_t \prod_{i=1}^{d} x(i)^{t(i)} \,\Big|\, a \in \mathbb{R}^{|I_d^k|} \}$$

<u>**למידה עם *feature maps*:**</u>

•ל-*feature maps* יכול להיות מימד גבוה, עבור $\mathcal{X} = \mathbb{R}^d$ אזי $|I_d^k| \approx d^k$ ונצטרך $\mathcal{F} = \mathbb{R}^n\,, n \approx d^k$.

שתי בעיות:
1. מימד גבוה מצריך הרבה דוגמאות בשביל לממש אלגוריתם *ERM* .
   פתרון – נשתמש באלגוריתם כמו *SVM*.

2. מימד גבוה מצריך חישובים כבדים והרבה זיכרון. (אפילו ייצוג הדוגמאות w דורש הרבה זיכרון)
   פתרון- שימוש בפונקציות *kernels* .

The Kernels Trick

נגדיר $K(x,x') := < \psi(x), \psi(x') >$ - פונקציה זו גם נקראת פונקציית *kernels* .

הטריק – אלגוריתמי למידה רבים עבור מפרידים ליניאריים יכולים להיות ממומשים באמצעות פונקציית K וללא ψ.

מתי נוכל להציג את w בלי לרשום את הקורדינטות?
דרך חלופית לייצג את w קיימת אם w היא *minimizer of an objective of a the following form*:

$$Minimize_w f(< w, \psi(x_1) >, \dots, < w, \psi(x_n) >) + R(||w||)$$

כאשר הפונקציה f היא: $f: R^m \to R \cup \{\infty\}$.

טענה: אלגוריתם *Hard SVM , Soft SVM* הם מהצורה שתוארה למעלה ביחס לψ.

Theorem (The representer theorem)

For any objective of the form above, there is a solution w of the form:

$$w = \sum_{i=1}^{m} \alpha(i)\psi(x_i), \qquad \text{where } \alpha = (\alpha(1), \dots, \alpha(m)) \in \mathbb{R}^m.$$

אם $n \gg m$ זה שיפור עצום.
עבור *hard-SVM* , w ניתן לייצוג עם $\alpha_i \neq 0$. במילים אחרות, w הוא צירוף ליניארי של ה- *support vector* .

בעיית *SOFT-SVM* באמצעות שימוש בפונקציות קרנל :

Kernel Soft-SVM

input The training sample Gram matrix $G \in \mathbb{R}^{m \times m}$, the training labels $y_1, \dots, y_m$, parameter $\lambda > 0$.
output $\alpha \in \mathbb{R}^m$
1: Find α that solves the following problem:

$$\text{Minimize}_{\alpha,\xi} \lambda \alpha^T G \alpha + \frac{1}{m}\sum_{i=1}^{m} \xi_i$$
$$\text{s.t. } \forall i, y_i \langle \alpha, G[i] \rangle \geq 1 - \xi_i, \text{ and } \xi_i \geq 0.$$

($G[i]$ is row i of G)
2: Return α.

•הערך המוחזר מהאלגוריתם הוא α. נוכל להבין מי זה w כך : $w = \Sigma_{i=1}^{m} \alpha(i)\psi(x_i)$.

•ההיפותזה היא $h_w(\psi(x)) = sign(< w, \psi(x) >) = sign(\Sigma_{j=1}^{m} \alpha(j) K(x_j, x))$.

•על מנת לחשב $K(x_j, x)$ נצטרך לשמור בזיכרון את כל הדוגמאות . (אלא אם כן $\alpha(j) = 0$).

נושא: *Polynomial kernel*

The k-degree polynomial kernel is $K(x,x') := (1 + \langle x, x' \rangle)^k$. $\quad (x, x' \in \mathbb{R}^d)$

טענה: זו פונקציית *kernel* . כלומר, קיימת פונקציה ψ שמתאימה לפונקציית קרנל שהגדרנו.

•אחת הפונקציות היא : $\psi: \mathbb{R} \to \mathbb{R}^{|l_d^k|}$ כך שכל קורדינטה בווקטור $\psi(x)$ הוא $\sqrt{B(k,t)} \prod_{i=1}^{d} x(i)^{t(i)}$.

סיבוכיות :

•ללא שימוש בקרנל - $O(d^k)$.
•עם שימוש בפונקציית קרנל - $O(d * \log(k))$.

נושא: *Gaussian kernel*

The Gaussian kernel with a parameter $\sigma > 0$ is $K(x, x') = e^{-\frac{\|x-x'\|^2}{2\sigma}}$. $\quad (x, x' \in \mathbb{R}^d)$

טענה: זו פונקציית *kernel* . כלומר, קיימת פונקציה ψ שמתאימה לפונקציית קרנל שהגדרנו.

•אחת הפונקציות היא : $\forall n \in \mathbb{N}$ $\psi: \mathbb{R} \to \mathbb{R}^{|I_d^n|}$ כך שכל קורדינטה בווקטור $\psi(x)$ הוא $e^{-\frac{\|x\|}{2\sigma}} \sqrt{\frac{B(n,t)}{\sigma^n n!}} \prod_{i=1}^{d} x(i)^{t(i)}$.

$$h_w(\psi(x)) = \text{sign}(\langle w, \psi(x)\rangle) = \text{sign}(\sum_{i=1}^{m} \alpha(i) K(x_i, x)) = \text{sign}(\sum_{i=1}^{m} \alpha(i) e^{-\frac{\|x-x_i\|^2}{2\sigma}}).$$

נושא: *Using kernels to encode a hypothesis class*

•אם יש לנו ידע קודם על הבעיה, ייתכן ואנחנו יודעים איזו מחלקת היפותזות הכי תתאים ולכן נרצה להתאים פונקציית קרנל לפי מחלקת ההיפותזות הזו.

נושא: *Gradient Descent*

•הרבה בעיות למידה ניתנות לכתיבה על ידי בעיית מינימיזציה.

בצורה כללית נרשום : $Minimize_{w \in \mathcal{B}}\ R(w) + \ell(w, S)$

כך ש- $\mathcal{B} \coloneqq minimization\ domain\ , R \coloneqq regularization\ term\ , \ell \coloneqq loss\ function$

למשל עבור אלגוריתם *ERM* :

$$\mathcal{B} = \mathbb{R}^d, R(w) = 0\ , \ell(w, S) = err(h_w, S)$$

נושא: קמירות

Definition: Convex Sets

A set $C \subseteq \mathbb{R}^n$ is convex if for any two vectors $u, v \in C$, the straight line between them is also in C:

$$\forall \alpha \in [0, 1], \quad \alpha u + (1 - \alpha) v \in C.$$

•דוגמה לקבוצות קמורות : $\mathbb{R}^n, line\ in\ \mathbb{R}^n$.

Definition: Convex Functions

Let C be a convex set. Let $f : C \to \mathbb{R}$ be a function. f is convex if for every $u, v \in C$, $\alpha \in [0, 1]$,

$$f(\alpha u + (1 - \alpha) v) \leq \alpha f(u) + (1 - \alpha) f(v).$$

Theorem

If $C \subseteq \mathbb{R}^n$ is convex and $f : C \to \mathbb{R}$ is convex,
then every local minimum of f is also a global minimum.

טענה: פונקציות ליניאריות הן קמורות.

טענה: צירוף קוני (מקדמים אי שליליים) של פונקציות קמורות הוא גם קמור.

$$g(u) = \Sigma_{i=1}^{k} a_i f_i(u) \ \ for\ a_i \geq 0$$

טענה: מקסימום בין שתי פונקציות קמורות היא פונקציה קמורה.

•אלגוריתם *Hard-SVM* הוא בעיה קמורה מכיוון שהקבוצה $\mathcal{B}$ קמורה וכי $f(w) = ||w||^2$ היא בעיה קמורה.

•אלגוריתם *Soft-SVM* היא בעיה קמורה מכיוון שהקבוצה $\mathcal{B}$ קמורה וכי הפונקציה $f(w) = \lambda||w||^2 + \ell(w,S)$ היא קמורה.

•עבור בעיות קמורות- ניתן לפתור אותן בצורה יעילה ע"י *GD* .

הרעיון של *GD* :
1. נתחיל מ-w כלשהו ניחוש התחלתי.
2. כל עוד לא הגענו להתכנסות – נזיז את w בכיוון ירידה של $f(w)$.
3. ניקח ממוצע מכמה wים מהאיטרציות האחרונות של הלולאה.

מהו כיוון ירידה: $-\nabla f(w) := -2w$.

Gradient Descent

input Number of iterations T, step size $\eta > 0$
output $w \in \mathbb{R}^d$ that (approximately) solves $\mathrm{Minimize}_{w\in\mathbb{R}^d} f(w)$
1: $w^{(1)} \leftarrow (0,\dots,0)$.
2: **for** $t = 1 : T$ **do**
3: $\quad w^{(t+1)} \leftarrow w^{(t)} - \eta\nabla f(w^{(t)})$.
4: **end for**
5: Return $\bar{w} = \frac{1}{T}\sum_{t=1}^{T} w^{(t)}$.

אם התחום $\mathcal{B} \subset \mathbb{R}^n$ אז ייתכן כי אחת הריצות "יוציאו" את w מהתחום ולכן נצטרך לבצע הטלה להכניס את w לתוך התחום.

Theorem

Let $M = \min_{w\in\mathbb{R}^d} f(w)$. If f is a convex function, and $\eta := \frac{c}{\sqrt{T}}$ ($c > 0$ depends on f), then

$$f(\bar{w}) - M \le O(\frac{1}{\sqrt{T}}).$$

הגדרה: *sub-gradient* הוא $v \in \mathbb{R}^n$ עבור $f: C \to \mathbb{R}$ אם $\forall u \in C, f(u) \ge f(w) + < v, u - w >$

•במקרה של פונקציות קמורות, *sub-gradient* תמיד קיים.
•עבור פונקציות קמורות, ייתכן כי הן לא יהיו גזירות בנקודה מסוימת w ולכן משתמשים ב- *sub-gradient* על מנת להשתמש בכל זאת באלגוריתם של *GD* (שדורש גרדיאנט)
•*sub-gradient* של פונקציה f בנקודה w הוא כיוון שתמיד יהיה מתחת ל- $f(w)$.

•סימון: קבוצת ה-*sub -gradient* בנקודה w תסומן בתור $\partial f(w)$.

•כאשר קיים הגרדיאנט של f אזי $\partial f(w) = \{\nabla f(w)\}$.
•נוכל להשתמש בכל $v \in \partial f(w)$ באלגוריתם של *GD* במקום $\nabla f(w)$.

משפט: w ממזער את f אם ורק אם $(0,\dots,0) \in \partial f(w)$.

Subgradients for maximum functions

Let $f(w) := \max\{g_1(w), g_2(w)\}$, where g_1, g_2 are convex differentiable.
If $f(w) = g_i(w)$, then $\nabla g_i(w) \in \partial f(w)$.

עבור אלגוריתם *soft-SVM* :

$$v_i(w) := \begin{cases} 0 & y_i\langle w, x_i\rangle \ge 1 \\ -y_i x_i & \text{otherwise} \end{cases}$$

- Set $v_w := 2\lambda w + \frac{1}{m}\sum_{i=1}^{m} v_i(w)$. Use instead of $\nabla f(w)$ in GD algorithm.

נושא: Stochastic Gradient Descent

מבחינה חישובית – ל-*GD* נדרש $\Omega(m)$ לכל איטרציה. ננסה להשתמש ב*SGD* .

רעיון: בכל איטרציה נעריך את $\nabla\ell(w,S)$.

נגריל דוגמה אחת מ-*S* ונחשב את ה-*hinge loss* עליה במקום לבצע ממוצע מכל הדוגמאות.

Stochastic Gradient Descent

input Number of iterations T, step size $\eta > 0$
output $w \in \mathbb{R}^d$ that (approximately) solves $\mathrm{Minimize}_{w\in\mathbb{R}^d} R(w) + \ell(w,S)$

```
w^(1) ← (0, ..., 0).
for t = 1 : T do
    Draw a random i uniformly from {1, ..., m}
    w^(t+1) ← w^(t) − η(∇R(w^(t)) + ∇ℓ(w^(t), (x_i, y_i))).
end for
Return w̄ = (1/T) Σ_{t=1}^{T} w^(t).
```

Theorem

Let $M = \min_{w\in\mathbb{R}^d} f(w)$, where $f(w) = R(w) + \ell(w,S)$. If f is a convex function, and the step size is set to $\eta = \frac{c}{\sqrt{T}}$ ($c > 0$ depends on f), then

$$\mathbb{E}[f(\bar{w})] - M \le O(\frac{1}{\sqrt{T}}).$$

•אותה הבטחה על התכנסות אבל רק במקרה הצפוי.

•כל איטרציה $O(1)$ במקום $O(m)$.

•עבור *soft-SVM* , גודל הצעד שנותן הבטחות טובות הוא $\eta_t = \frac{1}{\lambda * t}$.

Stochastic Gradient Descent on a stream of examples

```
w^(1) ← (0, ..., 0).
while true do
    Get next sample (x_t, y_t)
    w^(t+1) ← w^(t) − η_t(∇R(w^(t)) + ∇ℓ(w^(t), (x_t, y_t))).
end while
```

•במקרה זה ניתן לעבוד עם מדגם אינסופי.
•אין צורך לאחסן דוגמאות.

Theorem

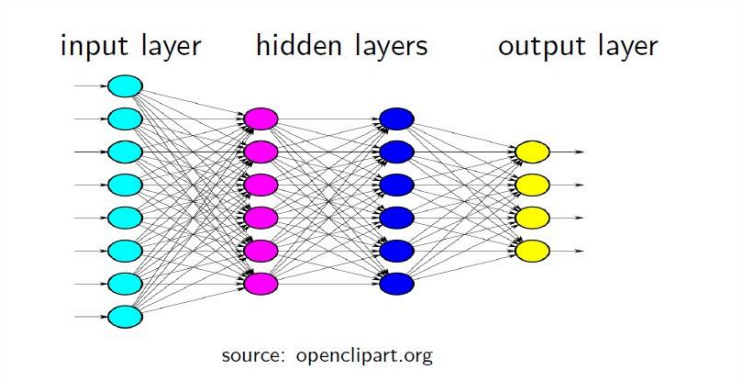
Let $M = \min_{w\in\mathbb{R}^d} f(w)$, where $f(w) = R(w) + \ell(w,\mathcal{D})$. If f is a convex function, and the step size is set to $\eta_t = O(\frac{c}{t})$, then at time t,

$$\mathbb{E}[f(\bar{w})] - M \le O(\frac{1}{\sqrt{t}}).$$

נושא: רשתות נוירונים

הגדרה: רשת נוירונים ניתנת לייצוג ע"י גרף מכוון $G = (V, E)$ כאשר הקודקודים V הם הנוירונים והצלעות מכוונות.

•אנחנו נדון ב-*feed-forward* שזה גרפים מכוונים חסרי מעגלים.
•לרוב אנחנו מארגנים את הגרף בשכבות לשם הנוחות.



הגדרה: רשת *feed-forward* שהיא ***fully-connected*** צריכה לקיים :

$$(i,j) \in E \; iff \; i \; and \; j \; are \;\; in \; consecutive \; layers$$

הגדרה: **קלט** הנוירונים הם הנוירונים שאין להם קשתות נכנסות.
כל קלט נוירון מהווה קורדינטה אחת בדוגמה. אם הדוגמה היא $x \in \mathbb{R}^d$ אז יהיו בשכבת הקלט d נוירונים.

הגדרה: **פלט** הנוירונים הם הנוירונים שאין להם קשתות יוצאות.
עבור סיווג בינארי של הדוגמאות, ניתן לעבוד עם פלט נוירון יחיד.
עבור k סיווגים של הדוגמאות, נעבוד עם k פלט נוירונים.

הגדרה: כל הנוירונים שאינם פלט או קלט נקראים נוירונים **חבויים**.

הגדרה: w הוא וקטור פרמטרים $w \in \mathbb{R}^{|E|}$.
w מפרט את המשקולות של כל אחת מהצלעות ברשת.

הגדרה: פונקציית אקטיבציה $\sigma: \mathbb{R} \to \mathbb{R}$. היא פונקציה ממשית לא-ליניארית. מטרת פונקציית האקטיבציה היא לתת מחלקת היפותזות גדולה ועשירה יותר.
פונקציות לדוגמה :
1. $\sigma(a) = sign(a)$.
2. $sigmoid: \; \sigma(a) = \frac{1}{1+\exp(-a)}$.
3. $hyperbolic \; tangence$: $\sigma(a) = \tanh(a) = \frac{\exp(2a)-1}{\exp(2a)+1}$.
4. $ReLU$: $\sigma(a) = \max(0, a)$.

הגדרה: פונקציית הסיווג מוגדרת על פי G (גרף הרשת) , פונקציית האקטיבציה σ, ו-וקטור הפרמטרים w .
והיא הפונקציה שבהינתן דוגמה $x \in \mathbb{R}^d$ מחזירה את התווית.

$$f_w^{G,\sigma}: \mathbb{R}^d \to y$$

החישוב שהפונקציה מחשבת נבנה מחישוב על נוירון אחד והחישוב מוגדר שכבה אחרי שכבה.
•כל נוירון בכל שכבה פולט את תוצאת החישוב שלו על בסיס משקל הקשת בין הנוירונים בין השכבות.
•קלט נוירון פולט כ-"תוצאת החישוב" את הקורדינטות של דוגמת הקלט x . כלומר, $o_i = x(i)$.
•הפלט של נוירון j בשכבות חבויות הוא $o_j := \sigma(\Sigma_{i:(i,j)\in E} w_{i,j} * o_i)$.

הגדרה: פלט של הרשת הוא וקטור k מימדי של מספרים ממשיים. (k הוא מספר הנוירוני פלט ברשת).

הגדרה: תרגום הפלט מתבצע בכך שלוקחים וקטור k מימדי שהוא פלט של רשת ומחשבים על בסיסו את התווית שתואמת את הדוגמה . תרגום פלט לדוגמה : *sign*.

הגדרה: ארכיטקטורת רשת מוגדרת על פי הפרמטרים G, σ, ψ.

הגדרה: מחלקת ההיפותזות של רשת נוירונים מוגדרת להיות $\mathcal{H}_{G,\sigma} = \{f_w^{G,\sigma} \,|w \in \mathbb{R}^{|E|}\}$.

אלגוריתם למידה של רשת :
קלט – גרף G ופונקציה σ.
פלט – וקטור פרמטר $\hat{w}$.

•עבור מחלקה זו אין אלגוריתמים *ERM* מוצלחים. אפילו לא עבור נוירון יחיד. ואפילו לא במקרה ה- *realizable* .
•נבחר פונקציית מטרה למזער. מכיוון שלמזער את $err(f_w, S)$ לכל w זה לא גזיר ננסה למזער את פונקציית ההפסד .

פונקציית ההפסד:

הדוגמה היא $x \in \mathbb{R}^d$ ו-k הוא מספר נוירוני הפלט. נגדיר את $g_w(x) = (o_1, \dots, o_k) = z \in \mathbb{R}^k$.

אזי פונקציית ההפסד תוגדר להיות $\tilde{\ell} : \mathbb{R}^k x\, Y \to \mathbb{R}$.
פונקציית ההפסד צריכה להיות גזירה על פי z .

$\tilde{\ell}(z, y)$ ימדוד את האי-התאמה בין הפלט של הרשת z לבין y . הערך של פונקציית ההפסד צריך להיות קטן רק כאשר $\psi(z)$ הוא התווית האמיתית.

עבור תוויות בינאריות ו – $k=1$ נוכל להגדיר את פונקציית ההפסד כך :

$squared\ loss := \tilde{\ell}(z, y) = (z - y)^2$.

ההפסד על המדגם יתואר באמצעות פונקציית ההפסד כך :

$\ell(w, S) = \Sigma_{i=1}^{m} \tilde{\ell}(g_w(x_i), y_i)$.

Stochastic Gradient Descent for Neural Networks

input Number of iterations T, step sizes $\eta_1, \eta_2, \dots > 0$
output $w \in \mathbb{R}^d$ that (approximately) locally minimizes $\ell(w, S)$
1: $w^{(1)} \leftarrow$ random, close to 0.
2: **for** $t = 1 : T$ **do**
3: Draw a random i uniformly from $\{1, \dots, m\}$
4: $w^{(t+1)} \leftarrow w^{(t)} - \eta_t \nabla \ell(w^{(t)}, (x_i, y_i))$.
5: **end for**
6: Return $w^{(t)}$ that works best on a validation set

•בשביל להריץ *SGD* , נצטרך לחשב את הגרדיאנט $\nabla_w \ell(w, (x, y))$ וזה נעשה בצורה של *back propagation* .
•נשתמש בכלל השרשרת ונקבל :

$$\nabla_w \ell(w, (x, y)) = \nabla_o \tilde{\ell}(o, y)|_{o=g_w(x)} * \nabla_w g_w(x)$$

מה קורה ברשת נוירונים עם $\tilde{\ell}$ כאשר התוויות הן לא בינאריות ?
נניח ו- $Y = \{1, \dots, k\}$ אזי $z \in \mathbb{R}^k$ הוא וקטור הפלט.

Squared loss:

עבור $z \in \mathbb{R}^k$ נגדיר $\tilde{\ell}(z, y) = \left|\left|z - e_y\right|\right|^2$.
במקרה זה נקבל כי $\nabla \tilde{\ell}(z, y) = 2(z^T - e_y)$.

Logistic / cross entropy loss:

לכל $z \in \mathbb{R}^k, i \in Y$ נגדיר $p_i(z) = \frac{e^{z(i)}}{\Sigma_{j=1}^{k} e^{z(j)}}$ זהו בעצם נרמול של הקורדינטות כך שסכומם יהיה שווה ל-1.
נגדיר את פונקציית ההפסד כך $\ell(z, y) = -\log(p_y(z))$.

$$for\ i \neq y : \nabla \ell(z,y) = p_y(z)$$
$$for\ i = y :\ \ell(z,y) = p_y(z) - 1$$
$$\bar{p}(z) = \big(p_1(z), \dots, p_k(z)\big) \coloneqq \boldsymbol{softmax}$$

מידת ההצלחה של *SGD* :

•הבעיה היא לא קמורה. אין הבטחות למצוא את המינימום הגלובאלי.

•בפועל, מצליחים בהרבה מקרים לקבל שגיאה 0 על המדגם. למה? זה לא ידוע. הצעות אפשריות:
-כאשר הרשת מאוד עמוקה, יהיו יותר מינימום גלובאליים ולכן קל למצוא אחד מהם.
- יש בעיות שמאוד קלות עבור חלק מהארכיטקטורות של הרשת.

•מה הופך את הבעיה להיות קלה יותר?
-מינימום לוקאלי קרובים בערכם למינימום גלובאלי.
-האתחול של המשקלים מובילים למינימום גלובאלי.
-הכיוונים של *GD* מוצלחים עבור בעיית הלמידה הזו.

•מחלקת היפותזות יותר עשירה ->יותר *overfitting* .

טענה: רשת נוירונים עם שכבה חבויה אחת ו- $\sigma = sign$ יכולה לבטח את כל הפונקציות הבינאריות.

טענה: מימד ה-*VC* של מחלקת ההיפותזות שמוגדרת ע"י רשת הנוירונים עם $\sigma = sign$ היא $O(|E|\log|E|)$.

•באמצעות רשת נוירונים ניתן להראות את כל הפונקציות ש"קלות לחישוב".

•נגדיר $\mathcal{F}_n$ להיות סט כל הפונקציות שניתנות לחישוב ע"י *n* צעדי חישוב במכונת טיורינג.

טענה: קיימת רשת נוירונים *G* בגודל $O(n^2)$ כך ש- $\mathcal{F}_n \subset \mathcal{H}_G$.

בתיאוריה- רשת יותר גדולה -> מחלקת היפותזות עשירה יותר -> יותר *overfitting* .
בפועל- אנחנו מקבלים פחות *overfitting* ככל שאנחנו מגדילים את השכבות החבויות ברשת.

Implicit regularization :
ברשת רחבה, *SGD* מעדיף לקחת פתרון עם נורמה נמוכה.

•יכול להיות גם ברשת נוירונים *overfitting* . למשל- סט אימון עם תוויות רנדומליות יכול לקבל שגיאה 0 על המדגם.

•על מנת שרשרת נוירונים תצליח בדרך כלל נצטרך מדגם אימון גדול.

•רשת נוירונים דורשת חישוב כבד על מנת ללמוד.
•רשת נוירונים דורשת חישוב כבד על מנת לסווג דוגמה.
•לפעמים מומלץ להשתמש בשיטות הפשוטות יותר.

נושא: עצי החלטה

הגדרה: עץ החלטה הוא פרדיקטור $h: X \to Y$ שמבצע החלטה ע"י ביצוע מעבר משורש העץ ועד העלים.

•העלים של העץ הם התוויות של העץ. כמה מהתוויות יכולות לחזור יותר מפעם אחת בעלים בעץ. (יש יותר מדרך אחת להגיע למסקנה הזו)
•החוליות הפנימיות של העץ מתאימות לפיצ'רים של דוגמה ב-*X* . ולכל חוליה כזאת יהיו ילדים שתואמים את התוצאות האפשריות .

מחלקת היפותזות עבור עצי החלטה:

עבור מקרה בינארי בסיסי בו $X = \{0,1\}^d, Y = \{0,1\}$ נוכל לקבל כל פונקציה אפשרית. ואנחנו עלולים לקבל *overfitting* .
נפתור את זה ע"י הגבלת גודל העץ.
$$|\mathcal{H}_n| \leq (d+2)^n$$
sample complexity : $O(n * \log(d))$.

ושוב – *trade off*:

• n גדול – *approximation error* קטן יותר , ו-*estimation error* גדול יותר.
• n קטן – *approximation error* גדול יותר , ו-*estimation error* קטן יותר.

ERM* על עצי החלטה זו בעיה *NP-hard ☹️

איך נפתור את זה? נמצא עץ עם שגיאה נמוכה על המדגם (לאו דווקא הכי נמוכה!) ובכך נקטין את השגיאה על ההתפלגות.

היוריסטיקה חמדנית *ID3*:

•נבחר את עץ ההחלטה בצורה חמדנית , החל מהשורש.
$Gain(S,i)$ היא פונקציה שמעריכה את השיפור על השגיאה של העץ על המדגם.
בכל איטרציה נחליט על איזה פיצ'ר i לפיו נשפר את העץ. נעצור כאשר העץ לא יכול עוד להשתפר על ידי הוספת פיצולים.

ID3 algorithm

input Training sample S, feature subset $A \subseteq \{1,\dots,d\}$.
output A tree for S using only attributes in A.
1: **if** all of S is labeled $y \in \{0,1\}$, return a leaf labeled y.
2: **if** $A = \emptyset$, return a leaf labeled with the majority label on S.
3: Let $j = \text{argmax}_{i \in A} \text{Gain}(S,i)$.
4: For $a \in \{0,1\}$, let $S_a = \{(x,y) \in S \mid x(j) = a\}$.
5: Return a tree with root j, left child $ID3(S_0, A \setminus \{j\})$, and right child $ID3(S_1, A \setminus \{j\})$.

פונקציית ה-*Gain*:
מעריכה את השיפור של הוספת פיצול i על השגיאה בעץ .
בפועל- לא ידוע העץ ולכן לא ניתן לחשב את השגיאה בעץ לכן נחשב את השגיאה במצב שזו נניח שהפיצול שביצענו הוא האחרון לפני התיוג, ונתייג את הדוגמאות לפי התווית הנפוצה בקרב כל תת קבוצה S_0, S_1.
סימון: q הוא השבר שמסמל כמה מתוך הקבוצה מכילה תיוג 1 .
$err(q) \coloneqq \min(q, 1-q)$.

$$err_{before}(S) \coloneqq err(q)$$

• בכל צד נקבל טעות שהיא $err(q)$. נחשב את השגיאה האמפירית :

$$err_{after}(S,i) = p_i^0 * err(q_i^0) + (1-p_i^0) * err(q_i^1)$$

ואז נקבל $Gain(S,i) = err_{before}(S) - err_{after}(S,i)$.

פונקציית ה-*Gain* הזו לא יודעת להבדיל בין שני פיצ'רים שמביאים אותה שגיאה למרות שאחד יכול להיות מועדף מבחינות אחרות. ננסה למצוא פונקציית *Gain* טובה יותר.

פונקציות ה-*Gain* הטובה יותר:

1. נחליף את השגיאה על q ב- $Entropy(q) \coloneqq -q * \log(q) - (1-q) * \log(1-q)$
2. נחליף את השגיאה על q ב- $Gini(q) \coloneqq 2q(1-q)$.

•הבעיה *NP* קשה ולכן אין הבטחות על אף אחת מהפונקציה. בפועל, משתמשים בפונקציות הטובות יותר.

האלגוריתם החמדני *ID3* לא מגביל את גודל העץ ולכן יש סיכוי שעבור גודל עץ מאוד גדול נקבל *overfitting* .
כדי למנוע בעיה זו, נבצע גיזום לעץ. נסיר חלקים מהעץ שעבורם לא נגדיל מאוד את השגיאה.

אופציות לגיזום:
1. החלפת תת עץ עם תווית 0 .
2. החלפת תת עץ עם תווית 1.
3. החלפת תת העץ עם התת עץ השמאלי שלו
4. החלפת תת העץ עם התת עץ הימני שלו.

עצים עבור דוגמאות בעלות ערכים ממשים :

יכולות להיות הרבה אופציות עבור שאלה לפיצ'ר.
threshold tests : עבור הפיצ'ר ה-*i* וסף θ נאפשר טסט $\mathbb{I}[x(i) \leq \theta]$ עבור מדגם *S* בגודל *m* יש לכל היותר *m+1* ספים אפשריים שיוצרים פיצולים שונים. נסמן אותם : $\theta_{1,i}, \dots, \theta_{m+1,i}$
נוכל להמיר כל פיצ'ר לבינארי ע"י המרה של פיצ'ר ממשי ל-*m+1* פיצ'רים בינאריים שהם : $\mathbb{I}[x(i) \leq \theta_{j,i}]$.

נריץ את האלגוריתם החמדני *ID3* עבור $(m+1)d$ פיצ'רים החדשים שיצרנו.

Random Forest :

פואנטה: היער יכיל מספר עצי החלטה שונים והפרדיקציה תקבע על ידי התווית הנפוצה שנקבל בקרב העצים ביער.

נצטרך ליצור מספר עצי החלטה על בסיס מדגם אימון יחיד. נבחר פיצ'רים בצורה רנדומלית, נלמד עץ החלטה על בסיס תת-קבוצה רנדומלית של סט האימון, נקבע את התווית של העלים בעץ לפי התווית הנפוצה ביותר בקרב התת-קבוצה.

•בתכלס- זה מאוד יעיל על הרבה בעיות ומשתמשים בזה הרבה.

נושא: Naïve Bayes

מטרה: נשתמש בתכונות מסוימות של ההתפלגות $\mathcal{D}$ כדי להקל על האלגוריתם למידה.

ההנחה של האלגוריתם :

$$Conditional\ independence\ of\ the\ features\ conditioned\ on\ the\ label$$

- **Independence**:
  A and B are independent ⇔
  knowing the value of A does not help to guess the value of B.
- **Conditional independence**:
  A and B are independent conditioned on Y ⇔
  when Y is known, knowing the value of A does not help to guess the value of B.

נניח כי $X = \{-1,1\}^d, Y = \{-1,1\}$ ונרצה לחשב את $h_{bayes}(x)$:

$$h_{bayes}(x) = argmax_{y \in Y}\ \mathbb{P}[Y = y] \prod_{i=1}^{d} \mathbb{P}[X(i) = x(i) | Y = y]$$

כמה פרמטרים אנחנו צריכים לדעת על מנת לדעת מהי $h_{bayes}(x)$ תחת ה-*naïve Bayes assumption* ? ***2d+1*** . מי הם :

$$p_0 = \mathbb{P}[Y = 1]$$
$$p_i := \mathbb{P}[X(i) = 1 | Y = 1]\ \ for\ 1 \leq i \leq d$$
$$p_i' := \mathbb{P}[X(i) = -1\ | Y = 1]\ for\ 1 \leq i \leq d$$

ונניח כי $p_i = p_i'$. כלומר – המקרים לטעות סימטריים. לכן נצטרך רק *d+1* פרמטרים.

לאחר הוצאת לוג וסידור הביטוי נקבל כי :

$$h_{bayes}(x) = sign(\Sigma_{i=1}^{d} \log\left(\frac{p_i}{1-p_i}\right) * x_i)$$

קיבלנו ש- $h_{bayes}(x) = sign(< w^*, x >)\ for\ w_i^* := \log\left(\frac{p_i}{1-p_i}\right)$ פרדיקטור ליניארי.

נותר לנו לחשב את ערכי ה-p_i :

איך נחשב p_0 ? נחשב את כמות המופעים של התווית 1 חלקי גודל המדגם.

איך נחשב p_i? נעבור על הקורדינטה ה-*i* בכל הדוגמאות ונבחן באיזה אחוז מהדוגמאות הקורדינטה ה-*i* תואמת לתווית הנכונה.

$$plug\ in\ decision\ rule : \hat{h}(x) = sign\left(\Sigma_{i=0}^{d}\widehat{w_i}x_i\right)\ for\ \widehat{w_i} = \log\frac{\widehat{p_i}}{1-\widehat{p_i}}\ , for\ \widehat{p_i} = \frac{k_i}{m}$$

$$k_i = number\ of\ correct\ examples\ for\ ith\ expert$$

כמה דוגמאות נצטרך?

עבור $m \geq \log\frac{\frac{2(d+1)}{\delta}}{2\epsilon^2}$ נקבל בהסתברות של לפחות $1-\delta$ עבור כל i : $|p_i - \widehat{p_i}| < \epsilon$.

נושא: Regression

מטרה: נתמודד עם בעיות עם תוויות לא בינאריות. $Y \subseteq \mathbb{R}$.

בעיות : 1. יש מספר אינסופי של תוויות
2. לא ניתן להבחין בין טעות קטנה לטעות גדולה.

נגדיר פונקציית הפסד חדשה: $\ell: YxY \to \mathbb{R}_+$.

$\ell(y,\hat{y})$ תגיד כמה גרוע התווית שחזינו $\hat{y}$ לעומת התווית האמיתית y .

פונקציות הפסד אפשריות: 1. $absolute\ loss : \ell_{abs} = |y-\hat{y}|$
2. $quared\ loss : \ell_{sq} = (y-\hat{y})^2$.

•לעומת מקרים אחרים שהשתמשנו בפונקציית הפסד במקום חישוב השגיאה על מנת לבצע חישובים בצורה נוחה, פה- פונקציית ההפסד היא המטרה האמיתית שאנחנו מעוניינים למזער.

$$\ell(h,\mathcal{D}) = \mathbb{E}_{(X,Y)\sim\mathcal{D}}[\ell(h(X),Y]$$

•על מנת להבין מיהי הפונקציה h_{bayes} נבחן את פונקציית ההפסד שבה אנחנו משתמשים.

נפתח את ההגדרה לפונקציית ה-*bayes*:

$$h_{bayes} = argmin_{h\in\mathbb{R}^x}\mathbb{E}[\ell(h(X),Y)]$$

Squared Loss:

. $h_{bayes}(x) = \mathbb{E}[Y|X=x]$

Absolute Loss:

. $h_{bayes}(x) = MEDIAN_{(X,Y)\sim\mathcal{D}}[Y|X=x]$

MEDIAN – הוא החציון.

הגדרה: $Empirical\ loss : \ell(h,S) = \frac{1}{m}\Sigma_{i=1}^{m}\ell(h(x_i),y_i)$.

•גם פה יש *bias-Complexity tradeoff*:

$$Approximation\ error : \ell_{app} := \inf_{h\in H}\ell(h,D)$$

$$Estimation\ error : \ell_{est} := \ell\left(\widehat{h_S},D\right) - \inf_{h\in H}\ell(h,D)$$

•מקרה שבו ניתן לבצע *ERM* יעיל הוא רגרסיה ליניארית.

הגדרה: רגרסיה ליניארית מחזירה פרדיקטור ליניארי עם פונקציית הפסד ריבועית.

פורמאלי:

- Instance space $\mathcal{X} = \mathbb{R}^d$
- Label space $\mathcal{Y} = \mathbb{R}$
- Hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$:
$$\mathcal{H} = \{h_{w,b} : \mathbb{R}^d \to \mathbb{R} \mid w \in \mathbb{R}^d, b \in \mathbb{R}\},$$
where $h_{w,b}(x) := \langle w, x\rangle + b$.
- Squared loss: $\ell_{\text{sq}}(y, y') = (y - y')^2$.

הבעיה בתור *ERM* : נמצא *w* שימזער את פונקציית ההפסד על המדגם. כלומר:

$$w \in argmin_{w\in\mathbb{R}^d}\Sigma_{i=1}^{m}(< w, x_i > -y_i)^2$$

רגרסיה ליניארית ידועה גם בשם ***Leased Squared*** (שלמדנו רק 7 פעמים ☹️)

איך פותרים *LS*?

1. פתרון באמצעות נוסחה סגורה ע"י המשוואות הנורמליות.
2. ניתן לפתור בשיטות איטרטיביות מכיוון שהבעיה קמורה ע"י *GD* או *SGD* .

נוכל להציג את ה-*m* דוגמאות שלנו במטריצה *X* בגודל *dxm* .

ואת התוויות של הדוגמאות בתור וקטור $y \in \mathbb{R}^m$.

<u>והמשוואה הנורמלית שנקבל היא :</u> $w := (XX^T)^{-1}Xy$. **במקרה בו XX^T הפיכה ו-*w* הוא פתרון יחיד.

במקרה בו XX^T לא הפיך, הוא המקרה בו יש מימד לא מנוצל. כלומר- יש פיצ'רים "מיותרים".

במקרה זה, נחליף את $(XX^T)^{-1}$ ב- $(XX^T)^{+}$. במקרה זה, *w* הוא אחד מהפתרונות למשוואה.

<u>הגדרה:</u> *pseudo inverse* : עבור מטריצה $A \in \mathbb{R}^{mxn}$ נגדיר את A^+ לקיים:
- $AA^+A = A\,,\ A^+AA^+ = A^+,\ (AA^+)^T = AA^+,\ (A^+A)^T = A^+A$.

אם A^TA הפיכה אזי $A^+ = (A^TA)^{-1}A^T$. אחרת- $A^+ = A^TA + \lambda I$. עבור $\lambda > 0$ המטריצה תמיד הפיכה.

<u>סיבוכיות חישוב:</u>

על מנת לחשב $(XX^T)^{+}$ נצטרך לבצע היפוך מטריצה $O(d^3)$.

סיבוכיות סך הכל תהיה : $O(md + d^3)$.

<u>*Sample complexity*:</u>

Theorem

Let $\mathcal{H} = \{h_w \mid w \in \mathbb{R}^d\}$, where $h_w(x) := \langle w, x\rangle$.
With a high probability over the sample $S \sim \mathcal{D}^m$, for all $h \in \mathcal{H}$,

$$\ell(h, \mathcal{D}) \le \ell(h, S) + O\left(\sqrt{\frac{d}{m}}\right).$$

סך הכל- *sample complexity* הוא $O(d)$.

מה נעשה אם *d* מאוד גדול? נגביל את מחלקת ההיפותזות ע"י הגבלת הנורמה של *w* .

במקרה זה, סך הכל נקבל *sample complexity* שהוא $O(B^4R^4)$.

Regularized LS:

•נשתמש בפונקציית עונש בה "נשלם" על נורמה גדולה במקום להגביל את הנורמה של w .

במקרה זה בעיית האופטימיזציה שלנו היא : $Minimize_{w\in\mathbb{R}^d}\lambda||w||^2+\Sigma_{i=1}^m(<w,x_i>-y_i)^2$

והפתרון לבעיה זו יהיה : $w=(XX^T+\lambda I)^{-1}Xy$.

•נבחין שבמקרה זה $\lambda>0$ והמטריצה תמיד תהיה הפיכה ולכן תמיד יתקבל פתרון יחיד למערכת.

Kernelization:

בבעיית ה-*Regularized LS* פונקציית המטרה היא :

$$f(w)=\lambda||w||^2+\Sigma_{i=1}^m(<w,\psi(x_i)>-y_i)^2$$

משפט הייצוג מתקיים כאן ולכן ניתן להשתמש ב*kernel trick* 😊

$$\alpha=(\lambda I+G)^{-1}y$$

*סיבוכיות חישוב הוא $O(m^3)$.

Theorem

Suppose training set $S=\{(x_i,y_i)\}_{i\le m}$. If we have:

- *Training points in ball of radius R in feature space:*

$$\forall i\le m,\quad \|\psi(x_i)\|=\sqrt{K(x_i,x_i)}\le R$$

- $\mathcal{H}=\{h_w(x)\equiv\langle w,x\rangle \mid \|w\|\le B\}$, *that is:*

$$w=\sum\alpha_i\psi(x_i) \text{ and } \|w\|=\sqrt{\alpha^\top G\alpha}\le B$$

- $|y_i|\le BR$ *for all* $i\le m$.

Then with high probability, for all $h\in\mathcal{H}$, $\ell(h,\mathcal{D})\le\ell(h,S)+O\left(\frac{B^2R^2}{\sqrt{m}}\right)$.

• רגוליזציית ℓ_2 בדרך כלל מוביל לפתרון w שמכיל הרבה קורדינטות שאינם אפסים . כלומר, w שהוא ***dense*** .
על מנת למנוע את זה נוכל להשתמש ברגוליזציית ℓ_1 : $||w||_1=\Sigma_{i=1}^d|w|$.

הגדרה: וקטור *sparse* הוא וקטור שמכיל הרבה קורדינטות בעלות ערך 0 .

חסרונות ויתרונות לשימוש ברגוליזציית ℓ_1:
1. אין נוסחה סגורה , חייב להשתמש בפתרון כמו *quadratic problem* . ☹️
2. לא תקף משפט הייצוג ולכן לא ניתן לפתור את הבעיה באמצעות *kernel trick* . ☹️
3. ניתן לחשב את כל הפתרונות עבור כל ערכי λ בפעם אחת. 😊

• *sample complexity* תלוי בכמות הקורדינטות שהם לא אפס יש בפתרון w .
• כאשר יש פתרון שהוא *sparse* יש פחות *overfitting* . 😊
•פונקציית המטרה בשימוש עם נורמת 1 נקראת *LASSO* .

נושא: PCA

•סוג של אלגוריתם למידה לא-מונחית . כלומר, מקבלים מדגם לא מתויג.

מציאת מבנה במידע- מוטיבציה:
•לעיתים המבנה הוא המטרה. למשל- מיון תמונות.
•לפעמים זה שלב ביניים לפני אלגוריתם למידה. מידע ממוין יכול להיות יותר נוח מבחינה חישובית (חיפוש במידע, וכו'). וגם מבחינה סטטיסטית, למידע מסודר יותר ניתן למצוא הכללה טובה יותר.

הגדרה: *PCA = Principal Component Analysis* זו טכניקה להורדת מימד.

מטרה: בהינתן מידע במימד $\mathbb{R}^d$, להוריד מימד ל- $\mathbb{R}^k$ בלי לאבד יותר מדיי מידע.

פורמאלי: בהינתן $(x_1, \dots x_m)$ נקודות במימד $\mathbb{R}^d$ ומימד היעד k צריך למצוא $U \in \mathbb{R}^{dxk}, V \in \mathbb{R}^{kxd}$

אשר ממזערות את פונקציית המטרה : $f(U,V) = \Sigma_{i=1}^{m} ||x_i - UVx_i||_2^2$.

U = "decompressor" , V = "compressor"

טענה: קיים פתרון אופטימלי לבעיה עם $U = V^T, U^T U = I_k$.

•קיבלנו כי $W^T x$ ממזער את פונקציית המטרה $f(u) = ||x - Wu||_2^2$.
למעשה קיבלנו $WW^T x$ היא הנודה הקרובה ביותר ל-x ב-R. מדובר בהטלה אורתוגונלית של x ב-R.

-W היא מטריצה המקיימת $W^T W = I_k$.

כעת ניתן לייצג את הבעיה של *PCA* בצורה הזו :

$$minimize_{U \in \mathbb{R}^{dxk}: U^T U = I_k} \Sigma_{i=1}^{m} ||x_i - UU^T x_i||_2^2$$

הגדרה: $trace(A)$ הוא הסכום של ערכי האלכסון של A עבור מטריצות ריבועיות.

•עבור מטריצות $trace(AB) = trace(BA)$: $A \in \mathbb{R}^{pxq}, B \in \mathbb{R}^{qxp}$.

ראשית נפשט את פונקציית המטרה ונקבל כי ניתן לייצג את הבעיה של *PCA* בצורה הזו:

$$maximize_{U \in \mathbb{R}^{dxk}: U^T U = I_k} trace(U^T \Sigma_{i=1}^{m} x_i x_i^T U)$$

טענה: הערך של הפתרון האופטימלי הוא לכל היותר $\Sigma_{i=1}^{k} \lambda_i$.

טענה: קיים פתרון אשר מביא את הערך $\Sigma_{i=1}^{k} \lambda_i$.

הפתרון אשר מביא את הערך האופטימלי הוא $\widehat{U} = [\widehat{u_1}, \dots, \widehat{u_k}]$ כאשר $\widehat{u_i}$ הם k הווקטורים העצמיים התואמים ל-k הע"ע הגדולים ביותר.

הפתרון עבור הבעיה המקורית של *PCA* הוא : $\Sigma_{i=k+1}^{d} \lambda_i$

•ניתן להציג את בעיית ה-*PCA* בצורה שונה ולהגיע לאותו פתרון. נבחר את ה-k מימדים אשר ממקסמים את השונות של האיברים במדגם.

סיבוכיות חישוב :

•חישוב מטריצה A הוא $O(md^2)$.

• לכסון מטריצה A יעלה $O(d^3)$.

עבור מקרים בהם *m>>d* נוכל להגדיר $B = XX^T$ כך ש- $B_{i,j} = < x_i, x_j >$ ונוכל ללכסן את B במקום את A ולקבל סיבוכיות חישוב של $O(m^3)$ ללכסן את B ולחשב את B ב- $O(m^2 d)$.

נושא: *Clustering*

פורמאלי: X הוא מרחב הדוגמאות.

$$metric\ p: XxX \to \mathbb{R}_+$$

*פונקציה p מקיימת אי שליליות, סימטריות, ואי שוויון המשולש.

אלגוריתם *Clustering* :

קלט : סט אימון S

פלט : חלוקה של S לקבוצות $C = (C_1, \dots C_K)$.

האלגוריתם:

נתחיל מחלוקה של S ליחידונים. בכל איטרציה נמזג את שני ה-*clusters* הכי קרובים.

"הכי קרובים" ייקבע ע"י פונקציית המרחק $p(A,B)$.

נעצור כאשר : א. כאשר נגיע ל-k קבוצות
ב. נעצור כאשר $p(A,B) > r$ – כאשר המרחק הבא שנצטרך למזג רחוקים מדיי.

- **Single Linkage** for $A, B \subseteq \mathcal{X}$
$$\rho(A,B) = \min\{\rho(x,x') : x \in A, x' \in B\}$$
- **Max/Complete Linkage** for $A, B \subseteq \mathcal{X}$
$$\rho(A,B) = \max\{\rho(x,x') : x \in A, x' \in B\}$$
- **Average Linkage** for $A, B \subseteq \mathcal{X}$
$$\rho(A,B) = \frac{1}{|A||B|}\sum_{x\in A, x'\in B}\rho(x,x')$$

גישה אפשרית אחרת עם *clustering* היא להגדיר פונקציית מטרה על *clusters* אפשריים ב-S.

$$f : C_S \to \mathbb{R}_+$$

אלגוריתם ל*Clustering* הוא לנסות למצוא *clustering* בצורה שממזערת את פונקציית העונש.

<u>אלגוריתם *K-mean*:</u>

k-mean זה אלגוריתם מסוג *objective-bayes* .

נגדיר את מרכז הקבוצה (*centroid*) להיות :

$$\mu(C_i) := argmin_{z\in X}\Sigma_{x\in C_i}p(x,z)^2$$

*ה-*centroid* לא חייב להיות חלק מהקבוצה S .

$$G_{\text{k-means}}(C_1,\dots,C_k) := \sum_{i=1}^{k}\sum_{x\in C_i}\rho(x,\mu(C_i))^2 = \min_{\{\mu_i\}_{i=1}^k\in\mathcal{X}}\sum_{i=1}^{k}\sum_{x\in C_i}\rho(x,\mu_i)^2.$$

בעיה זו היא *NP* קשה ולכן לא ניתן למצוא לה פתרון יעיל, לכן נפתור את הבעיה ע"י אלגוריתם איטרטיבי.

k-means algorithm for Euclidean space

input A data set $S \subseteq \mathbb{R}^d$; number of clusters k
output $\mu_1,\dots,\mu_k \in \mathbb{R}^d$
1: init random $\mu_1,\dots,\mu_k \in \mathbb{R}^d$ based on S
2: **repeat**
3: For every $i \in \{1,\dots,k\}$ define cluster C_i as
$$C_i := \{x \in S \mid i = \operatorname*{argmin}_{1\le j\le k}\|x-\mu_j\|\}.$$
4: For every $i \in \{1,\dots,k\}$ define the centroid of C_i as
$$\mu_i := \frac{1}{|C_i|}\sum_{x\in C_i}x.$$
5: **until** convergence (no change in cluster allocation)

•הבעיה מתכנסת אך לאו דווקא לפתרון האופטימלי.

k-mean++:

בחירת המרכזים הראשוניים בצורה חכמה ולא רנדומלית.

בתוחלת, אלגוריתם זה מקבל $O(\log k)$ פקטור מעל הפתרון האופטימלי.

גרסאות נוספות של אלגוריתמים מהסוג *center-based* :

- k-**medoids**: like k-means, centroids from dataset:

$$G(C_1,\ldots,C_k) = \min_{\mu_1,\ldots,\mu_k \in S} \sum_{i=1}^{k} \sum_{x \in C_i} \rho(x,\mu_i)^2$$

.1

באלגוריתם זה , המרכזים נבחרים רק מתוך ה*data* שיש לנו.

k-**medians**: like k-medoids, but not squared

$$G(C_1,\ldots,C_k) = \min_{\mu_1,\ldots,\mu_k \in S} \sum_{i=1}^{k} \sum_{x \in C_i} \rho(x,\mu_i)$$

.2

גם באלגוריתם זה, המרכזים נבחרים מתוך ה-*data* שיש לנו אך פה פונקציית המרחק היא לא בריבוע.

•אלגוריתמים מסוג זה ניתנים להתבצע בשיטה האיטרטיבית כמו ה-*kmeans* .

Clustering Axioms

נגדיר תכונות שהיינו רוצים שיהיה לאלגוריתם.

1. *Scale Invariance* :

$$\forall S \subset X\,, \rho: XxX \to \mathbb{R}_+\,, \alpha > 0: \quad F(S,\rho) = F(S,\alpha\rho)$$

סקלאר בפונקציית מרחק לא ישפיע על חלוקת ה*clusters* .

2. *Richness* :

לכל תת קבוצה S של X ולכל חלוקה חוקית C של S ולכל פונקציית מרחק, כל החלוקות האפשריות יוכלו להתקבל כפתרון לאלגוריתם $F(S,\rho) = C$.

3. *Consistency* :

יהי נקודות *x,y* כך ששניהם שייכים לאותו *cluster* .

ונגדיר ρ' :

$$\forall x,y \in S\ s.t: x = y\,, \rho'(x,y) \le \rho(x,y)\ and$$

$$\forall x,y \in S\ s.t: x \ne y\,, \rho'(x,y) \ge \rho(x,y)$$

נקבל $F(S,\rho) = F(S,\rho')$. (אותו פתרון של חלוקה C עבור פונקציית המרחק ρ'.

מטרה: למצוא פונקציה שתקיים את התכונות.

בעיה: לא קיימת פונקציה שמקיימת את כל שלוש התכונות בו זמנית. ☹️

Theorem

There is no clustering function F satisfying all axioms (SI), (Ri), (Co).

ניתן להראות בקלות שקיימים פונקציות שמקיימות 2/3 מהתכונות.

נושא: *Generative models*

נניח שיש לנו משפחה של התפלגויות $X: \{\mathcal{D}_\theta \mid \theta \in \Theta\}$, $\theta = (\mu, \sigma) \in \mathbb{R}^2$ ו- $\mathcal{D}_\theta = \mathcal{N}(\mu, \sigma^2)$.

מטרה: למצוא θ^* בהינתן $S \sim \mathcal{D}_\theta^m$.

תכונות עבור ה-*estimators* :

1. חסר הטיה- כלומר - $\mathbb{E}_{S\sim\mathcal{D}_\theta^m}[\hat{\theta}] = \theta$.
2. עקביות - $\forall\epsilon, \lim_{m\to\infty} P[|\hat{\theta} - \theta^*| \geq \epsilon] = 0$. הכוונה היא שעבור מדגם ששואף לאינסוף, ה-*estimator* שלנו יתקרב ל*estimator* האופטימלי.

על מנת למצוא את $\hat{\theta}$ נשתמש בשיטת *MLE* (*Maximum likelihood estimator*)

$$\hat{\theta} = argmax_\theta \, L(S;\theta)$$

נפתור ע"י גזירה ומציאת נקודת מקסימום.

טריק חוזר : להוציא לוג ולחפש נקודת מקסימום ללוג. מטרה: מכפלה הופכת לסכום והכל יותר נחמד.

***Mixture Distributions* :**

ערבוב של כמה סוגי התפלגויות. במקרה זה, לעיתים קשה לגזור ולהשוות לאפס ולכן לא ניתן למצוא את המקסימום.

נבצע מקסימיזציה בצורה שונה ע"י שימוש במשתנים חבויים. נשתמש במשתנים חבויים ובשיטה איטרטיבית שדומה ל*kmeans* על מנת לחשוב בקלות את המקסימום.

***The Expectation-Maximization trick* :**

איך נשערך את ערכי ה-z_i שנדרשים (המשתנים החבויים).

EM for mixture of 2 Gaussians

input $(x_1, \ldots, x_m) \in \mathbb{R}^m$
output $(p, \mu_0, \mu_1) \in [0,1] \times \mathbb{R} \times \mathbb{R}$

1: $t \leftarrow 0$, select $(p^{(t)}, \mu_0^{(t)}, \mu_1^{(t)})$ randomly
2: **repeat**
3: E-step: pseudo-counts

$$q_i^{(t)} := \frac{p^{(t)} \frac{1}{\sqrt{2\pi}} e^{-(x_i-\mu_1^{(t)})^2/2}}{(1-p^{(t)}) \frac{1}{\sqrt{2\pi}} e^{-(x_i-\mu_0^{(t)})^2/2} + p^{(t)} \frac{1}{\sqrt{2\pi}} e^{-(x_i-\mu_1^{(t)})^2/2}}, \qquad i \leq m$$

4: M-step: maximize augmented likelihood

$$p^{(t+1)} := \frac{1}{m}\sum_{i=1}^m q_i^{(t)}; \qquad \mu_1^{(t+1)} := \frac{\sum_{i=1}^m q_i^{(t)} x_i}{\sum_{i=1}^m q_i^{(t)}}; \qquad \mu_0^{(t+1)} := \frac{\sum_{i=1}^m (1-q_i^{(t)}) x_i}{\sum_{i=1}^m (1-q_i^{(t)})}$$

5: $t := t+1$;
6: **until** convergence (no change in $L(S; p^{(t)}, \mu_0^{(t)}, \mu_1^{(t)})$)

- ההתכנסות של *EM* היא איטית.
- אם קשה לגזור את הביטוי אזי *gradient ascent* יהיה איטי .
- לא נדרש לבחור את גודל הצעד לעומת *gradient ascent*.
- עבור בעיות מעורבבות, *EM* כמעט תמיד יהיה מועדף על *GA*.
- ב-*EM* , אילוצים על פרמטרים נכפים כחלק מהאלגוריתם.