שיעור 1

שיעור 1 נקודות חשובות:

- כדי לאמן מערכת גדולה עם המון פרמטרים, צריך גם המון דוגמאות בהתאם.

  - שיטה אחת להתמודד עם מעט דוגמאות כאשר יש המון פרמטרים היא: transfer learnning – כלומר להתשמש ברשת שאומנה על דאטה סט גדול יחסית שידוע לזהות מזה תמונה טבעית או חפצים מסויימים בתמונה ואז להשתמש בפרמטרים שלה ולאמן את הרשת שלנו כדי לדייק אותם למטרה שלנו. (לחשוב על כך שיש לנו נקדות התחלה טובה יחסית במנוחי גרדיינט, ואנחנו נרצה להתקרב למינימום, אז עדיף שנתחיל מנקדות התחלה טובה ולא אקראית.)

- לרוב, ברשתות CNN השכבות הראשונות ברשת זהות כמעט לכל משימה כלשהי, אז ההבדל לרוב מגיע בשכבות האחרונות, לכן עוד הסבר למה transfer learnning עובד.. (כלומר נצטרך להחליף רק את השכבות האחרונות כדי להתאים את הרשת לבעיה שלנו!.)

- כדי שהשיטה תעובד, עבור Task A,B חייב להתקיים:

  - אותו **input** בשתי הבעיות. אם זה בתמונות אז אותו גודל וכו..
  - גם שיהיה איזה חפיפה בין מה הרשת הראשונה לומדת והשניה, לא חייב חפיפה גדולה.

- לרוב ב- CNN השכבות הראשונות לומדים features התחלתיים מהתמונה, כגון קווים או איזור מאפינים חשובים שיש כמעט בכל תמונה וממשכים כך עד שבונים כל התמונה בשכבה הסופית. וזה גם מחזק למה השכבות הראשונות מושתפות כמעט לכל בעיה כלשהי למשל בזהוי תמונות.

- כדי שתהיה למידה מוצלחת, אז אחד התנאים החושבים, זה שיש לכל הדוגמאות באימון התפלגות משותפת והשאיפה שדוגמה חדשה שתגיע שלא הייתה באימון תהיה מהתפלגות זהה, אחרת לא יהיה הכללה ולא יהיה למידה. במקרה הטוב יהיה שינון (overfiting).

- קצת יותר מדויק איך עושים transfer-learning:

  - **אופציה אחת**: עושים pre-training על דאטה גדול, לומדים\פותרים משימה A ואז מעתיקים את הרשת כמו שהיא ואת הפרמטרים שלה, ופותרים בעיה B. השכבות הראשונות לא נוגעים בהם, (לא מאמנים אותם, מעבירים את הקלט עד השכבות האחרונות) אם דאטה קטן אז מחליפים שכבה אחרונה בלבד אם יותר גדול, אז אפשר יותר משכבה ומאמנים רק אותם. שאר השכבות נמצאות ב- freeze.
  - **אופציה שנייה**: עושים אותו הדבר כמו אופציה אחת אבל ההבדל שמאמנים מחדש את כל הרשת (כל השכבות) אבל חשוב מאוד, לאמן למספר קטן של epcohs וגם עם lr נמוך, כי אחרת עלולים לדרוס את המשקולות שקיבלנו וגם אפילו להגיע ל- overfiiting.

- למידה Multitask: למשל אם רוצים לזהות עובר משתמש מסויים אם המיילים שהוא מקבל ספאם או לא, אז יש לו מעט דאטה וגם לא יעיל\חכם לאמן מודל ספיציפי לכל משתמש בנפרד. אז משתמשים בכל הדאטה שיש לכל המשתמשים ומאמנים מודל אחד שיזהה ספאם. ואז אפשר אחרי זה "לדייק" יותר את המודל כך שנאמן אותו על הדאטה סט של כל אחד מהמשתמים בנפרד אבל רק עבור השכבות האחרונות.

- בהינתן סט דוגמאות $\{(x_n, y_n)\}$, רשת ניורונים טובה תנסה ללמוד את הקשר $f(x_n)$ כמה שיותר טוב על סמך הדגימות שיש לנו. ואז בתקווה שנקבל דגימה חדשה x נוכל לחשב את $f(x)$ כמה שיותר טוב.

- בבעיות אי־ליניאריות צריך למפות את הנתונים למרחב עשיר יותר באמצעות ϕ. יש שלוש דרכים: להשתמש במפה כללית (עלול לקבל overfiting), להנדס ידנית תכונות (צריך מומחיות), או הגישה של למידה עמוקה – לתת למודל ללמוד בעצמו את פונקציות הבסיס. למרות שהלמידה אינה קמורה וקשה יותר לאופטימיזציה, היא מאפשרת לגלות מבנים לא־ליניאריים מורכבים ולכן היתרונות גדולים מהחסרונות.

שיעור 2

שיעור 2 נקודות חשובות:

- כוחה של רשת ניורונים לעומת מודל לינארי פשוט, הוא שאפשר לקרב כל קשר שנרצה בין הקלט לפלט בדיוק שנרצה. (מודל לינארי יכול לבטא\לקרב קשרים לינאריים).

- פונקציית מחיר הכי מעניינות כמו MSE או $CrossEntropy$ במודל לינארי הם $convex$ אך שמדובר ברשת ניורונים (עמוקה), זה לא המצב, הפונקציות האלו לא יהיו $convex$ בגלל האי-לינאריות של הרשת (פונקציות הפעלה של ה hidden layers).

- במודל לינארי, עבור פונקציות מחיר שהם convex יש התכנסות למינימום גלובלי בתיאוריה. אבל עבור רשת נירונים (עמוקה), לא נוכל לדעת ולא מובטח לנו בכלל התכנסות למינימום גלובלי אלא רק לוקלי (הסיכוי להגיע למינימום גלובלי הוא אפסי, אפילו גם קשה להגיע למינימום גלובלי).

- ברשתות ניורונים משתמשים בכללי בשיטות איטרטיביות מבוססות גרדיינט כדי למזער את פונקציית המחיר.

- כדי להפעיל שיטות גרדיינט, כמובן צריך לבחור פונקציית מחיר וגם איך לייצג את ה- output.

- יחידת ה- output עושה לנו טרנספורמציה סופית של כל התכונות בשכבה האחרונה כדי לסיים את המשימה ולקבל את התוצאה הנדרשת. ולכן בחירה של פונקצייה המחיר מאוד תלוייה ביחידת ה- output כלומר פוצקיית activation של ה- output.

- גישה maximum likelihood: אנחנו בוחרים את הפרמטרים שנותנים את ההסתברות הכי גבוהה לנתונים שראינו בפועל. למשל אם מתוך 10 הטלות של מטבע יצאו 7 עצים – ההסתברות הכי סבירה לעץ היא 0.7, כי זה הערך שנותן את ההסתברות הגבוהה ביותר לנתונים שראינו.

- ערך הפרמטרים θ שיביא למקסימום את $f_Y(y;\theta)$ הוא בדיוק כמו ערך הפרמטרים של $\log f_Y(y;\theta)$ כי $\log$ פונק' מונוטונית עולה. (משתמשים ב- $\log$ לרוב כי יש זהויות שמפשטות את החישבוית\מעברים המתמטיים).

- הגישה המקובלת לבניית פונקציית מחיר היא Maximum likelihood.

- אפשר להתייחס לפונקציה שהרשת מנסה ללמוד כ- $p(y \mid x;\theta)$ (לא חייב תמיד להיות ככה), כלומר פונקצייה שממפה x ל- y (פונקצייה ההסתברות המותנית של y בהיינתן x עם סט של פרמטרים θ). נרצה להפעיל גשית Maximum likelihood, כלומר למצוא θ שממקסם את $\log p(y \mid x;\theta)$ או בצורה שקולה למצוא θ שממזער את $-\log p(y \mid x;\theta)$. (negative log-likelihood).

- היתרון בגישה הזאת הוא: שאם נדע לקבוע את הפונקציית ה- output לבעייה שלנו, אז יש לנו מתכון איך לקבל פונקציית מחיר מתיאמה. (וכמובן גם לקבוע את מבנה הרשת וכו..).

- נרצה שהגרדיינט של פונקציית ה- loss להיות גדול. מכיוון שהגרדיינט הוא מנחה אותנו בלמידה שלנו. כלומר איך להשתפר עוד ובאיזה כיוון להתקדם כדי למזער את ה- loss, כי אם יהיה קטן (אפסי) אנחנו נהיה תקועים ולא נוכל להתקרב בכלל לשיפור או התכנסות.

- פוצקייה שמגיעה לרוויה, בנקודות האלו היא "שטוחה" כלומר בנוקדות האלו הגרדיינט שלה הוא אפסי וזה בעייתי עבורנו בלמידה מבוססת גרדיינט ואנחנו נרצה להימנע ממקרים כאלו. זה יקרה לנו (להימנע מגרדיינט אפסי) אם נשתמש בפונקציות activation (לרוב לא לינארית) ביחידות המצוא והחבויות.

- אז אם נשתמש בפונקציות אקטיבציה ביחידות המוצא או החביות שמיגיעות לרוויה, אז אנחנו עלולים להיתקע בבעיה של vanishing gradient (גרדיינט אפסי) ולכן נרצה לבחור כאלה שלא מגיעות לרוויה.

- עצם זה שמתשמים בפונק' מחיר שהיא negative log-likelihood תעזור לנו להתמודד במקרים של רוויה **במוצא** של הרשת.

- נגיע לרוויה כי לרוב בפונקציות מוצא יהיה לנו אקספוננט שמגיע לרוויה (כמעט 0) בערכים מאוד שליליים. ואז פונקציית ה- $\log$ מבטלת את האקספוננט וזה איך הגישה של negative log-likelihood תעזור לנו להתמודד עם vanishing gradient במוצא.

- פונקציות loss כמו MSE\MAE נקבל איתם ביצועים חלשים כאשר משתמשים בשיטות גרדיינט, חלק מהיחידות שלנו יגיעו לרוויה ואז נקבל ביצועים פחות טובים ולכן עוד סיבה למה להשתמהש ב- negative log-likelihood.

- בהינתן $\hat{y} = W^T h + B$, אז ניתן להניח ש- $p(y \mid x;\theta) = N(y;\hat{y}(x;\theta), I)$, כלומר מתפלג נורמלית עם תוחלת $\hat{y}(x;\theta)$ ושונות I (אפשר להניח שכולה אחדים) כלומר יש לנו את הפילוג של y בהינתן x. ואז נרצה להפעיל גישת maximum likelihood על $p(y \mid x;\theta)$, לקבל את θ שמסבירות הכי טוב את המדידות שיש לנו (כי התלות בין $\hat{y}$ ל- x זה סט הפרמטרים θ שיש לנו ברשת).

שימוש בגישת negative log-likelihood עבור מוצא ליניארי עם הסברים בין המעברים:

בהינתן $\hat{y} = f(x;\theta) = w^T h + b$ (כלומר יש מוצא ליניארי identity אחד בלבד) אז נניח כי $p(y \mid x;\theta) = N(y; f(x;\theta), \sigma^2)$ (מתפלג נורמלית ו- σ^2 ידועה). בנוסף נתון סט נתונים $\left\{(x_n, y_n)_{n=1}^N\right\}$ כאשר הדוגמאות הן מאותה התפלגות וגם קיים בניהן אי-תלות (i.i.d). אז נכתוב עובר נקודה x_n ספיציפית:

$$p(y_n \mid x_n;\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(y_n - f(x_n;\theta))^2}$$

נרשום את פונקציות הסבירות:

$$L(\theta) = p(y \mid x;\theta) \overset{i.i.d.}{=} \prod_{n=1}^{N} p(y_n \mid x_n;\theta) = \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(y_n - f(x_n;\theta))^2}$$

נכתוב את $\log L(\theta)$ ונקבל:

$$\begin{aligned}
\log L(\theta) &= \log\left(\prod_{n=1}^{N} p(y_n \mid x_n;\theta)\right) \overset{(1)}{=} \sum_{n=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(y_n - f(x_n;\theta))^2}\right) \\
&\overset{(1)}{=} \sum_{n=1}^{N} \log\left(\left(2\pi\sigma^2\right)^{-\frac{1}{2}}\right) + \log\left(e^{-\frac{1}{2\sigma^2}(y_n - f(x_n;\theta))^2}\right) \\
&\overset{(2)+(3)}{=} \sum_{n=1}^{N} \underbrace{-\frac{1}{2}\log\left(2\pi\sigma^2\right)}_{constant} - \frac{1}{2\sigma^2}\left(y_n - f(x_n;\theta)\right)^2 \\
&= -\frac{N}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{n=1}^{N}\left(y_n - f(x_n;\theta)\right)^2
\end{aligned}$$

כעת נרצה למקסם את:

$$-\left(\frac{1}{2\sigma^2}\sum_{n=1}^{N}\left(y_n - f(x_n;\theta)\right)^2 + \frac{N}{2}\log\left(2\pi\sigma^2\right)\right)$$

או למזער את **מינוס** אותו הבטוי, כלומר למזער את:

$$\frac{1}{2\sigma^2}\sum_{n=1}^{N}\left(y_n - f(x_n;\theta)\right)^2 + \frac{N}{2}\log\left(2\pi\sigma^2\right)$$

כיוון שרוצים למזער $-\log L(\theta)$, וגם כי $\frac{N}{2}\log(2\pi\sigma^2)$ לא תלוי ב- θ, אז מספיק למזער (בצורה שקלוה)

$$J(\theta) = \frac{1}{N}\sum_{n=1}^{N}\left(y_n - f(x_n;\theta)\right)^2$$

החלפנו את $\frac{1}{2\sigma^2}$ ב- $\frac{1}{N}$ וזה לא משנה את θ שיביא למינימום את $J(\theta)$, פשוט נקבל ממוצע בהחלפה הזאת וכדי לקבל את הצורה של MSE (וגם מחלקים בגדול הדאטה שפשוט לא ישפיע על החישוב). אז לסיכום אנחנו נרצה למזער את $J(\theta)$, כלומר למצוא $\hat{\theta}$ כך ש-

$$\boxed{\hat{\theta} = arg\min_{\theta}\frac{1}{N}\sum_{n=1}^{N}\left(y_n - f(x_n;\theta)\right)^2}$$

$$\log(x\cdot y) = \log x + \log y \quad (1)$$
$$\log a^b = b\log a \quad (2)$$
$$\log e^x = x \quad (3)$$

לסיכום: אחרי שקבענו את פונקציה המוצא להיות לינארית כלומר identity, והפעלנו תהליך negative log-likelihood קיבלנו שפונקצית ה- loss המתאימה כאן היא MSE. (כנל אפשר לעשות עבור sigmoid במקום identity ולראות שפונקצית ההפסד שמתקבלת היא Cross-Entropy).

הערה: אמנם השכבה הסופית היא לינארית אך כל הרשת בין הקלט לפלט היא לא ליניארית ולכן $\hat{\theta}$ מביא למינימום את פונקצית המחיר כאשר היא לא לינאירת וגם לא convex.

מסקנה: ברגע שיש לנו פונקצית אקטיבציה לינאירת **במוצא** אז זה "הכי נוח" לנו מבחינת אלגורתמי גרדינט כי אין לנו את העניין של vanishing-gradient במוצא. (אבל זה לא אומר דבר או חצי דבר על מה שיקרה בשכבות החבויות, יכול להיות בעיות של vanishing-gradient גם שם.. רמז: צריך גם שם פונקציות הפעלה מתאימה אבל זה כבר בשיעור הבא!).

בעיית binary classifiton: כלומר עכשיו $\hat{y} = 0, 1$.

- נרצה לייצר רשת כאשר המוצא של הרשת הוא ההסתברות $p(y=1 \mid x)$, וכמובן התוצאה נופלת בטווח $[0,1]$.
  - כדי להשתמש במוצא הזה לסיווג, אז למשל אפשר להחליט כי אם:

$$p(y=1 \mid x) \geq t \rightarrow \hat{y} = 1$$
$$p(y=1 \mid x) < t \rightarrow \hat{y} = 0$$

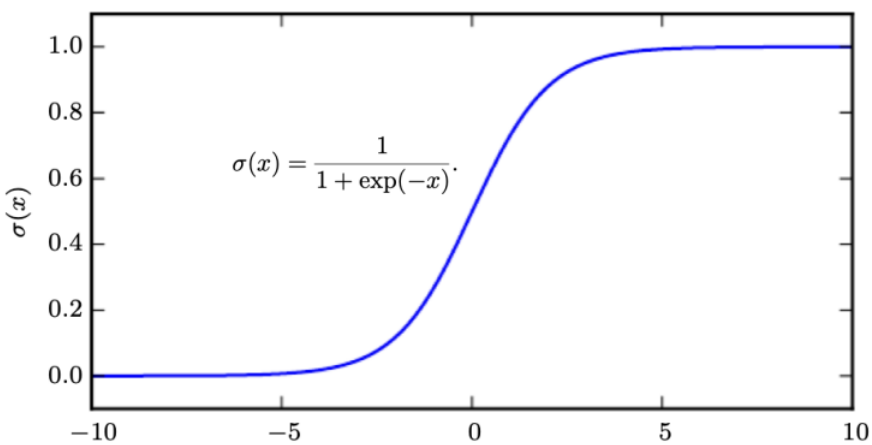
למשל אפשר לבחור $t = 0.5$.

- אפשר להציע את פונקציית ההפעלה הבאה: $p(y=1 \mid x) = \max\left(0, \min\left(1, w^T h + b\right)\right) \in [0,1]$ אבל זה מאוד בעייתי מבחינת למידה בעזרת מורד הגרדיינט כי למשל עבור כל ערך של

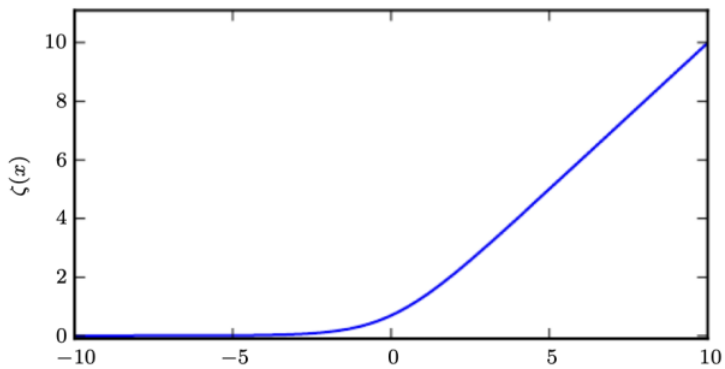
$w^T h + b$ שהוא גדול או קטן מאחד נקבל גרדיינט אפס ואז אין יכולת למידה ואנחנו מעדיפים פונקציית ההפעלה שיש לה גרדיינטים גדולים במיוחד כאשר הערכים הם שגויים.

- פתרון אחר לנרמול בין $[0,1]$ זה פונקציית הפעלה sigmoid $\sigma(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$. ובשילוב עם גישת ה- maximum likelihood תמיד נקבל גרדיינט חזק כאשר המודל שוגה.

- תכונות של sigmoid:
  - $\sigma(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}} \in [0,1]$
  - $x = 0,\ \sigma(x) = \frac{e^x}{1+e^x} = \frac{1}{2},\ x \to -\infty,\ \frac{1}{1+e^{-x}} \to 0$,$x \to \infty,\ \frac{1}{1+e^{-x}} \to 1$
  - $1 - \sigma(x) = 1 - \frac{e^x}{1+e^x} \overset{\text{מכנה משותף}}{=} \frac{1}{1+e^x} = \sigma(-x)$
  - $\frac{d\sigma(x)}{dx} = d\frac{\left(1+e^{-x}\right)^{-1}}{dx} = -\left(1+e^{-x}\right)^{-2}\left(-e^{-x}\right) = \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} = \sigma(x) \cdot (1 - \sigma(x))$
  - ואז נקבל גם $\frac{d\sigma(x)}{dx} = \sigma(x) \cdot (1 - \sigma(x)) = \sigma(x) \cdot \sigma(-x)$
  - בנקודות מסוימות של $\sigma(x)$, למשל $x \geq 5$ או $x \leq -5$ הגרדיינט כעמט אפסי (אפשר לראות לפי הגרף באיזה איזור הוא יותר שטוח)



- פונקציית Softplus $\zeta(x) = \log(1+e^x)$ והתכונות שלה:
  - לכל x נקבל $\zeta(x) = \log(1+e^x) \geq 0$
  - $\zeta(x) = \log(1+e^x) = -\log\left((1+e^x)^{-1}\right) = -\log\left(\frac{1}{1+e^x}\right) = -\log(\sigma(-x))$
  - $\frac{d}{d\zeta(x)} = \frac{d\log(1+e^x)}{dx} = \frac{e^x}{1+e^x} = \sigma(x)$
  - כאשר x מספיק גדול אז $\zeta(x) = \log(1+e^x)$ יתנהג כמו $\log(e^x) = x$ כלומר לינארי.



- בגישה של maximum likelihood $-\log\left(p\left(y \mid x;\theta\right)\right)$ ה- log מבטלה את ה- exp של ה-sigmoid ובלי זה, כמו שאמרנו הפונקצייה תוכל להגיע לרוויה בערכים מסויימים של x וזה ישפים על הלימוד בגישות גרדיינט. (לכן שילוב של פונקציות הפעלה sigmoid במוצא + גישת mll נפתור את הבעיה של vanishing gradient במוצא).

שימוש בגישת negative log-likelihood עבור מוצא sigmoid עם הסברים בין המעברים:

בהינתן $\hat{y} = f\left(x;\theta\right) = \sigma\left(w^T h + b\right)$ (sigmoid), סט נתונים $\left\{\left(x_n, y_n\right)_{n=1}^{N}\right\}$ כאשר הדוגמאות הן מאותה התפלגות וגם קיים בניהן אי-תלות (i.i.d). אז נכתוב עובר נקודה x_n ספיציפית:

$$p\left(y_n \mid x_n;\theta\right) = f\left(x;\theta\right)^{y_n} \cdot \left(1 - f\left(x;\theta\right)\right)^{1-y_n}$$

בנוסף ידוע כי $y_n \in \{0,1\}$ אז ניתן לרשום:

$$p\left(y_n \mid x_n;\theta\right) = \begin{cases} f\left(x;\theta\right) & y_n = 1 \\ 1 - f\left(x;\theta\right) & y_n = 0 \end{cases}$$

נכתוב את הפילוג המשותף:

$$p\left(y \mid x;\theta\right) \overset{i.i.d.}{=} \prod_{n=1}^{N} p\left(y_n \mid x_n;\theta\right) = \prod_{n=1}^{N} f\left(x;\theta\right)^{y_n} \cdot \left(1 - f\left(x;\theta\right)\right)^{1-y_n}$$

נכוב את $-\log p\left(y \mid x;\theta\right)$ ונקבל:

$$\begin{aligned} -\log p\left(y \mid x;\theta\right) &= -\log\left(\prod_{n=1}^{N} f\left(x;\theta\right)^{y_n} \cdot \left(1 - f\left(x;\theta\right)\right)^{1-y_n}\right) \\ &\overset{(1)+(2)}{=} -\sum_{n=1}^{N} y_n \log\left(f\left(x;\theta\right)\right) + \left(1 - y_n\right)\log\left(1 - f\left(x;\theta\right)\right) \end{aligned}$$

לפני שנמשיך, ניתן לראות שקיבלנו כבר את Cross-Entropy

נסמן $f\left(x_n;\theta\right)=\sigma\left(w^Th+b\right)=\sigma\left(z_n\right)$ ונקבל

$$\begin{aligned}J\left(\theta\right)&=-\sum_{n=1}^{N}y_n\log\left(\sigma\left(z_n\right)\right)+\left(1-y_n\right)\log\left(\overbrace{1-\sigma\left(z_n\right)}^{\sigma(-z_n)}\right)\\&=-\sum_{n=1}^{N}y_n\log\left(\sigma\left(z_n\right)\right)+\left(1-y_n\right)\log\left(\sigma\left(-z_n\right)\right)\end{aligned}$$

ואפשר לרשום $J\left(\theta\right)$ כ-

$$J\left(\theta\right)=-\sum_{n=1}^{N}\log\left(\sigma\left(\left(2y_n-1\right)z_n\right)\right)=\sum_{n=1}^{N}-\log\left(\sigma\left(-\left(1-2y_n\right)z_n\right)\right)$$

ולפי הקשר $\zeta\left(x\right)=-\log\left(\sigma\left(-x\right)\right)$ נכתוב סופית:

$$\boxed{J\left(\theta\right)=\sum_{n=1}^{N}\zeta\left(\left(1-2y_n\right)z_n\right)}$$

וזו פונקציית המחיר שלנו במקרה של פונק' הפעלה sigmoid במוצא.

- עבור $\zeta\left(x\right)$ כאשר x מאוד שלילי נקבל ערכים אפס ואז יש בעיות של גרדיינט אפסי, עם זאת במקרה שלנו אם נרצה ש- $x=\left(1-2y_n\right)z_n$ יהיה מאוד שלילי, אז צריך להתקיים אחד משני המקרים:
  - $z_n=w^Th+b$ גדול מאוד חיובי ($\sigma\left(z_n\right)\approx 1$) ו- $\left(1-2y_n\right)$ שלילי, כלומר $y_n=1$ וזה אומר סיווג נכון ולכן אין בעיה שהגרדינט יהיה אפסי במקרה זה כי לא נרצה לפשר משהו נכון.
  - $z_n=w^Th+b$ גדול מאוד שלילי ($\sigma\left(z_n\right)\approx 0$) ו- $\left(1-2y_n\right)$ חיבוי, כלומר $y_n=0$ וזה אומר סיווג נכון ולכן אין בעיה שהגרדינט יהיה אפסי במקרה זה כי לא נרצה לפשר משהו נכון.
- עבור $\zeta\left(x\right)$ כאשר x מספיק גדול (לכיוון החיובי) נקבל ש- $\zeta\left(x\right)\approx x$ ולכן נקבל גרדיינט גדול מספיק במקרים שנדרשים תיקון.

לסיכום: במצבים שנדרשים תיקון הגרדיינט יהיה גבוה וההפך נכון.

- כאשר משתמשים בפונקציות הפסד אחרון למשל כמו MSE עם פונקציית הפעלה $\sigma\left(z\right)$, אז פונקציה ההפסד תוכל להגיע לרוויה בכל מקרה $\sigma\left(z\right)$ מגיעה לרוויה ולכן שילוב נכון הוא $\sigma\left(z\right)$ + maximum likelihood (כי הוא מבטל אל ה- exp).
- סיכום כל השיעור במשפט קצר: כדי להתמודד עם בעיות רוויה של פונקציית הפסד שגורמת לגרדיינט אפסי לרוב גישת ה- negative log-likelihood היא פתרון להמון מודלים.

שיעור 3

שיעור 3 נקודות חשובות:

- נכליל את sigmoid ל- softmax, כלומר במוצא יש לנו k תוצאות, וקטור בגודל k. אז:
  - הוקטור $\hat{y}$ יש בו k ערכים, כאשר כל ערך i מציג את $\hat{y}_i = p(y = i \mid x)$
  - נרצה שכל ערך בוקטור יהיה בתחום $[0, 1]$ וגם כן שהסכום של כל ערכי הוקטור יהיה 1 כי פשוט זה יתן לנו הסתברות לכל המחלקות.
  - כדי לעשות זאת, בהינתן $z = W^T h + b$ (וקטור המוצא), נגדיר: $z_i = \log(\widetilde{p}(y = i \mid x))$ ערך הסתברות לא מנורמל (z_i יכול להיות בין $(-\infty, \infty)$), כעת נרצה לנרמל אותו לערכים בין $[0, 1]$ וכך שגם סכום כל הערכים יהיה 1 על ידי:

$$softmax(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

  $\exp(z_i)$ יבטל לנו את הערכים השליליים ו $\sum_j \exp(z_j)$ ינרמל אותו לטווח בין $[0, 1]$. כיוון שאנחנו מחלקים כל ערך בסכום הזה, אז נקבל שסכום כל הערכים בוקטור שווה בדיוק ל- 1 כנדרש.

- במילים פשוטות, עבור דוגמה מסוימת, ה-softmax מחזיר את ההסתברות שהדוגמה שייכת לכל אחת מן המחלקות. (ובשלב החיזוי טבעי לבחור במחלקה עם ההתסברות הכי גבוהה)

- גם כאן כמו ב- $sigmoid$, גישת ה- maximum log-likelihood תעזור לנו כי פשוט תבטל לנו את ה- exp וככה נמנעים מבעיות רוויה בפונקציה. נקבל אחרי הפעלה ה- log:

$$\log softmax(z)_i = z_i - \log \sum_j \exp(z_j)$$

  z_i כבר ליניארי ואין שום בעיה של רוויה.

- כאשר יש הבדל בין הערכים של z אז ניתן לרשום את: $\log\left(\sum_j \exp(z_j)\right) \approx \max_j(z_j)$ כי הערך הכי גדול יהיה הכי משפיע בסכום בגלל ה- exp ואז כאילו ויש לנו $\log(\exp(\max_j(z_j)))$ ואז נקבל מה שכתוב למעלה. ולכן על סמך זה, אפשר לרשום כקירוב טוב:

$$-\log softmax(z)_i = \max_j(z_j) - z_i$$

וזה מלמד אותנו, שאם הלוגיט של המחלקה הנכונה z_i קרוב או שווה לערך המקסימלי בוקטור, השגיאה כמעט אפסית – ולכן המודל כמעט לא נענש. לעומת זאת, אם z_i רחוק מהערך הגדול ביותר (כלומר יש טעות), השגיאה גדלה והמודל "מעניש" חזק יותר. במילים אחרות: ללא טעות הגרדיאנט כמעט אפסי, ובעת טעות הגרדיאנט גדול.

- ב- softmax כמו ב-sigmoid אפשר להגיע רוויה. ב- sigmoid ראינו בשני צדדי הפונקציה ניתן להגיע לרוויה אך ב-softmax זה יקרה כאשר יש הפרשים גדולים בין ערכי הוקטור.

- כמו ב- sigmoid גם כאן לא כל פונקציית הפסד תעבוד טוב. אנחנו נרצה גישה שתבטל לנו את ה- exp כמו ה- negative log-likelihood.

- אפשר לרשום את ה- softmax בצורה הבאה:

$$softmax\left(z\right)_i = \frac{\exp\left(z_i\right)}{\sum_j \exp\left(z_j\right)} = \frac{\exp\left(c\right)\cdot\exp\left(z_i - c\right)}{\exp\left(c\right)\cdot\sum_j \exp\left(z_j - c\right)} = \frac{\exp\left(z_i - c\right)}{\sum_j \exp\left(z_j - c\right)} = softmax\left(z - c\right)_i$$

ואז מה שנהוג לעשות זה:

$$softmax\left(z\right) = softmax\left(z - \max_j\left(z_j\right)\right) = \frac{\exp\left(z - \max_j\left(z_j\right)\right)}{\sum_j \exp\left(z - \max_j\left(z_j\right)\right)}$$

מהחישוב הזה $z - \max_j\left(z_j\right)$ נקבל שכל הערכים קטנים או שווים ל-0, ולכן ה- $\exp\left(z - \max_j\left(z_j\right)\right)$ יתן לנו ערכים קטנים או שווים לאחד וחסומים על ידי 0. ובמילים אחרות, כך נמנעים מבעיות של overflow כי לחשב exp של מספר גדול זה בעייתי או פשוט לא אפשרי.

- ה- softmax יכול להגיע לרוויה ב-1 כאשר הקלט $z_i = \max_j\left(z_j\right)$ וגם z_i הרבה יותר גדול משאר הערכים בוקטור. גם יכול להגיע לרוויה ב-0 כאשר z_i הוא לא מקסימלי והוא הרבה יותר קטן מהרעך המקסימלי. (נראה בהמשך שפונקצית הפסד טובה מתמודדת טוב עם המצב הזה, פשוט במצבים אלו יהיה לנו סיווג טוב ואז גרדיינט אפסי דווקא טוב שאין שגיאה בחיזוי.)

פיתוח פונקציית הפסד ל- softmax:

נתון סט אימון: $i.i.d.$ $\left\{\left(x_n, y_n\right)_{n=1}^N\right\}$, $y_n = 1, 2, \ldots, k$, נרשום פונקצייה ה- likelihood:

$$L\left(\theta\right) = p\left(y \mid \left\{x_1, \ldots, x_N\right\}; \theta\right) \overset{i.i.d.}{=} \prod_{n=1}^N p\left(y_n \mid x_n; \theta\right)$$

ניתן לרשום את: $p\left(y_n \mid x_n; \theta\right)$:

$$p\left(y_n \mid x_n; \theta\right) = \prod_{k=1}^K \left(p\left(y_n = k \mid x_n; \theta\right)\right)^{I\left(y_n = k\right)}$$

$I\left(y_n = k\right)$ תחזיר 1 כאשר $y_n = k$ ואחרת 0. ומכאן נסיק שקיים איבר אחד במכפלה שרשמנו למעלה. נציב ונקבל:

$$L\left(\theta\right) = \prod_{n=1}^N \prod_{k=1}^K \left(p\left(y_n = k \mid x_n; \theta\right)\right)^{I\left(y_n = k\right)}$$

עתה נרשום:

$$J(\theta) = -\log L(\theta) = -\log \prod_{n=1}^{N} \prod_{k=1}^{K} \left(p\left(y_n = k \mid x_n; \theta\right)\right)^{I(y_n=k)}$$

ואז לפי חוקי לוג (1) + (2) נקבל:

$$J(\theta) = -\sum_{n=1}^{N} \sum_{k=1}^{K} I\left(y_n = k\right) \log\left(p\left(y_n = k \mid x_n; \theta\right)\right)$$

מכיוון ש- $y_n = 1, 2, \ldots, K$, ואין משעמות לסדר בניהם, אז ניתן להחליף את כל y_n בוקטור k ממדי, כך שיש 1 רק במחלקה אליה הוא שייך והשאר אפסים. למשל $y_n = 3$ יהפוך ל- $y_n = (0, 0, 1, 0, \ldots, 0)$.
ואז ניתן להחליף את $I\left(y_n = k\right)$ ב- y_{nk} כך שלכל דוגמה y_n יש לה רק k אחד בלבד שיתן 1 והשאר יתאפס. אז נכתוב:

$$J(\theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} -y_{nk} \log\left(\hat{y}_{nk}\right)$$

והגענו בדיוק ל- Cross-Entropy רב מחלקתי.
ואז לפי $\log softmax\left(z\right)_i = z_i - \log \sum_j \exp\left(z_j\right)$ נקבל:

$$J(\theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} -y_{nk} \log\left(\hat{y}_{nk}\right) = \sum_{n=1}^{N} \sum_{k=1}^{K} -y_{nk} \left(z_{nk} - \log\left(\sum_{j=1}^{K} \exp\left(z_{nj}\right)\right)\right)$$

ומכיוון שבסכום $\sum_{k=1}^{K} -y_{nk} \log\left(\hat{y}_{nk}\right)$ יש רק איבר אחד והשאר מתאפסים, אז ניתן לרשום את זה כ:

$$\sum_{k=1}^{K} -y_{nk} \log\left(\hat{y}_{nk}\right) = -\log\left(\hat{y}_{nc}\right)$$

כאשר c הוא הערך שאם נציב $k = c$ נקבל $y_{nk} = 1$. ומכאן אנחנו רואים, שהמודל מעניש רק על המחלקה הרלוונטית לחיזוי ולא כל המחלקות.
אז אם:

- $\hat{y}_{nc}$ מאוד גבוה, אז ההסתברות מתקרבת ל- 1 (חיזוי נכון) ו- log של משהו קרוב לאחד קרוב מאוד לאפס ולכן לא נעניש על מצב כזה. (כי הגרדיינט כאן יהיה קטן)
- $\hat{y}_{nc}$ מאוד נמוך, אז ההסתברות מתקרבת ל- 0 (חיזוי שגוי) ו- log של משהו קרוב לאפס ישאף למינוס אינסוף ואז כן נעניש על מצב כזה (כי הגרדיינט כאן יהיה גדול).

מסקנה: דוגמה שמסווגת הכי לא נכון (הסתברות הכי נמוכה) היא זאת שתניע אותנו הכי טוב בשיפור המודל ולהפך, דוגמה שמסווגת הכי טוב כמעט ולא תתרום למודל (לא תשפר אותו ולא תהרוס).

אז מכל זה נובע שמודל טוב ישאף לדחוף את כל הדוגמאות להסתברות גבוהה ככל האפשר עבור המחלקה הנכונה.

נראה שקילות בין sigmod ל- softmax כאשר $k=2$:

מההגדרה של softmax נקבל:

$$\hat{y} = f(x;\theta) = \begin{bmatrix} (y=1 \mid x;\theta) \\ (y=2 \mid x;\theta) \end{bmatrix} = \begin{bmatrix} \frac{\exp(z_1)}{\sum_{j=1}^{2}\exp(z_j)} \\ \frac{\exp(z_2)}{\sum_{j=1}^{2}\exp(z_j)} \end{bmatrix}$$

נסמן את $z_i = w_i^T h$ עבור $i=1,2$ ונקבל:

$$\begin{bmatrix} \frac{\exp\left(w_1^T h\right)}{\exp\left(w_1^T h\right)+\exp\left(w_2^T h\right)} \\ \frac{\exp\left(w_2^T h\right)}{\exp\left(w_1^T h\right)+\exp\left(w_2^T h\right)} \end{bmatrix} \overset{(*)}{=} \begin{bmatrix} \frac{1}{1+\exp\left((w_2-w_1)^T h\right)} \\ \frac{\exp\left((w_2-w_1)^T h\right)}{1+\exp\left((w_2-w_1)^T h\right)} \end{bmatrix} = \begin{bmatrix} \frac{1}{1+\exp\left(-(w_1-w_2)^T h\right)} \\ \frac{\exp\left(-(w_1-w_2)^T h\right)}{1+\exp\left(-(w_1-w_2)^T h\right)} \end{bmatrix}$$

$$= \begin{bmatrix} sigmoid\left(\left(-(w_1-w_2)^T h\right)\right) \\ 1-sigmoid\left(\left(-(w_1-w_2)^T h\right)\right) \end{bmatrix}$$

וכך קיבלנו sigmoid. מכיוון שסכום הערכים בוקטור הוא 1 אז מן הסתם נצטרך להחזיק $k-1$ ערכים בלבד בוקטור מכיוון שהערך הזה פשוט יהיה הערך המשלים לשאר ההסתבריות.

$(*)$ חילקנו מונה ומכנה ב- $\exp\left(w_1^T h\right)$.

גזירת פונקציית ה- $Loss$:

$$\frac{\partial J(\theta)}{\partial z_{ni}} = (p_{ni} - y_{ni})$$

כאשר p_{ni} היא ההסתברות שהדוגמה n שייכת למחלקה i ו- y_{ni} ה- label, 1 אם דוגמה n באמת היא ממחלקה i ואפס אחרת.

<u>נסתכל על מקרים של שיפור ה- Loss:</u>

- אם המחלקה i היא הנכונה, אז $y_{ni}=1$ ואז אם p_{ni} נמוך מאוד, נקבל משהו שקרוב למינוס אחד ולפי נוסחת הגרדיינט (אנחנו נוסיף מינוס הגרדיינט) ולכן נתקדם בכיוון החיובי ובמילים אחרות נגדיל את ההסתברות למחלקה הנכונה.

- נניח ש- i אינה המחלקה הנכונה, אז $y_{ni}=0$ ואז אם p_{ni} גדול מאוד (קרוב לאחד) נקבל משהו שקרוב לאחד ואז מסיבות דומות נתקדם בכיוון השלילי ואז נקטין את ההסתברות למחלקה הלא נכונה.

ואז הגרדיינט לשאר הפרמטרים כאשר $z = W^T h$ כתוב בצורה בקוטורית (כלל שרשרת):

$$\frac{\partial J(\theta)}{\partial z} = (p-y), \quad \frac{\partial J(\theta)}{\partial W} = (p-y)\,h^T, \quad \frac{\partial J(\theta)}{\partial h} = W(p-y)$$

שכבות חבויות

- כדי לקבוע פונקציות הפעלה ליחידות בשכבות החבויות, אין לנו איזה מתכון כמו שהיה ליחידת המוצא, פשוט ניסוי וטעיה ובוחרים מה שעובד יותר טוב.
- בכללי נדרוש שפונקציות אקטבציה יהיו גזירות, כי אנחנו מחשבים גרדיינט. זה גם בסדר גם אם הפונקציה תהיה לא גזירה במספר סופי של נקודות.
- אנחנו נרצה למזער את הגרדיינט ולא בהכרח להגיע לאיזה מינימום לוקלי, אז לא יפריע לנו שיש נקודות בהם הגרדיינט לא מוגדר.
- גם לרוב בפונקציות אקטבציה שנבחר, תהיה נקודה לא גזירה אבל כן קיימת הנגזרת שלה מימין ומשמאל לנקודה ולכן פשוט אפשר להחזיר את הנגזרת שלה מכיוון מסוים בנקודה הלא מודגרת.
- אפשר לבחור לכל שכבה חבויה פונקצית אקטבציה שונה, אבל לרוב לא עושים את זה, ומה שמקובל זה לבחור פונקצית אקטבציה אחת לכל השכבות החבויות.
- פונקצית הפעלה הכי פשוטה היא פונקציה ליניארית, כלומר מעבירים את הקלט איך שהוא ללא שינוי.
  - אם נשתמש בפונקציה מסוג זה, לא תהיה לנו משמעות לעומק של הרשת והרשת תהיה שקולה למודל ליניארי. כלומר אפשר לייצג אותה על ידי רשת עם שכבה אחת.
  - אם קיימת רשת עמוקה, כך שצוואר הבקבוק שלה (מספר היחידות בשכה חבויה מסויימת) קטן מהמינמום בין שכבת הקלט לפלט (במפסר יחידות) אז כאן יש מצב של הורדת מימד ואין יכולת ביטוי של רשת בעלת שכבה אחת.
  - אפשר לחסוך בפרמטרים בגישה הזאת, למשל אם יש N inputs M hidden untis ו- K outputs ללא bias אז יש לנו $M(N+K)$ פרמטרים, לעומת מודל לינארי (שכבה אחת) יהיה לנו NK פרמטרים, ובמקרים מסויימים ש- M קטן יחסית ביחס ל N, K אז נקבל ש- $M(N+K)$ קטן מ- NK.
- פוקנציות פשוטות שהתמשו בהן כפונקציות אקטבציה: sigmoid ו- tanh כאשר $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \in [-1, 1]$ וגם מתקיים: $tan(x) = 2 \cdot sigmoid(2x) - 1$ (פונקציות גזירות בכל נקודה)
  - כמו שראינו, גם ה-sigmoid וגם ה-tanh יש להם רוויה! והרוויה היא כמעט בכל התחום של הפונקציות. וזה בעייתי מבחנת הגרדיינט, כי בקטעים אלה הוא אפסי וזה הופך את הלמידה באמצעות גרדיינטים למאוד קשה.
  - במצוא היה לנו שיטה לבטל את הבעיתות של הרוויה, אבל בשכבות חבויות אין לנו את זה.
- בכללי, אם חייבים לבחור בין $tanh$ ל- $sigmoid$ אז $tanh$ עדיף מהסיבה שסביב האפס הוא מדמה פונקציה לינאירת (במונחי אינפי סביב 0, $tanh(x)$ מתנהג כמו x) ואז יש לנו גרדיינט יותר גדול.

- ב- $softplus\left(x\right)=\ln\left(1+e^{x}\right)$ יש לנו רק בערכים השליליים רוויה ולא בחיוביים, ובערכים חיוביים $softplus\left(x\right)$ מתנהג כמו x.

- המנצח שלנו יהיה $ReLU\left(x\right)=\max\left(0,x\right)$. מאוד נוח להשתמש בה כי היא מזכירה מאוד פונקציה לינארית, ופונקציה לינארית קל לנו לעבוד איתה שעובדים עם גרדיינטים מכיוון שמקבלים גרדיינטים גדולים איתם ואז יש יכולת לימדה טובה.

  - $ReLU\left(x\right)$ ניצח גם את $tanh$ ו- $sigmoid$ מבחינת תוצאות וגם מבחינת זמני ריצה. למשל ברשתות כמו ImageNet. (כי גם הפונקציה פשוטה מאוד וגם הנגזרת שלה גם פשוטה מאוד).
  - יש חסרון ל- ReLU במקומות הוא מאפס דברים (בערכים השליליים) אין שם יכולת למידה והגרדיינט יהיה אפס.
  - פתרון לבעיה הזאת, היא שיפורים לפונקציית ה- ReLU נקרא Leaky ReLU: $g\left(x\right)=\max\left(0,x\right)+\alpha\min\left(0,z\right),\ \alpha\in\left(0,1\right)$. לרוב בוחרים $\alpha=0.01$ וכך פותרים את הבעיה שהוא מאפס ערכים שליליים ואז שם הגרדיינט לא מתאפס.
  - אם נבחר $\alpha=-1$ אז נקבל ש- $g\left(x\right)=\left|x\right|$ לרוב משתמשים בזה בבעיות של עיבוד תמונה\ראייה מומחשבת ששם אין חשיבות לסימן.
  - אפשר לבחור α_i כפרמטר נלמד גם.

- בכללי נעדיף לאתחל את b להיות חיובי קטן כדי שהגרדיינטים לא יתאפסו לנו.

- $softplus$ מזכיר מאוד את $ReLU$ אבל בגרסה "חלקה" יותר וגם הוא גזיר בכל הנקודות וגם יותר טוב מבחינת רוויה! עם זאת, הוכיחו ש- $ReLU$ ניצח ואיתו קיבלנו תוצאות יותר טובות וזה מראה שקשה באמת לבחור פונקציה אקטבציה טובה לפי האינטוציה ולכן תמיד נרצה לבחון את התוצאות של פונקציות אקטבציה.

שיעור 4

שיעור 4 נקודות חשובות:

- בהינתן ε קטן מספיק, אז מתקיים: $f\left(x+\varepsilon\right)\approx f\left(x\right)+\varepsilon f'\left(x\right)$, אינטוציה לזה, כדי להגיע ל- $f\left(x+\varepsilon\right)$ ע"י $f\left(x\right)$, אז נצטרך את הכיוון הנכון בו אנחנו נתקדם + גודל הצעד. אז "טבעי" לבחור גודל צעד ε והכיוון יהיה השיפוע בסביבת הנקודה x כי ε קטן ולכן נבחר $f'\left(x\right)$.

- הנגזרת $f'\left(x\right)$ תתן לנו את כיוון העליה והירידה של הפונקציה ואז ניתן לדעת ביאזה כיוון נוכל למזער או להגדיל את הפונקציה.

  - עבור $sign\left(f'\left(x\right)\right)>0$ אם נלך ימינה $\left(+\right)$ בצעד ε ערך הפונקציה יגדל, ולכן אם נרצה למזער את ערך הפונקציה במקרה זה אז נלך בכיוון ההפוך של הנגזרת.

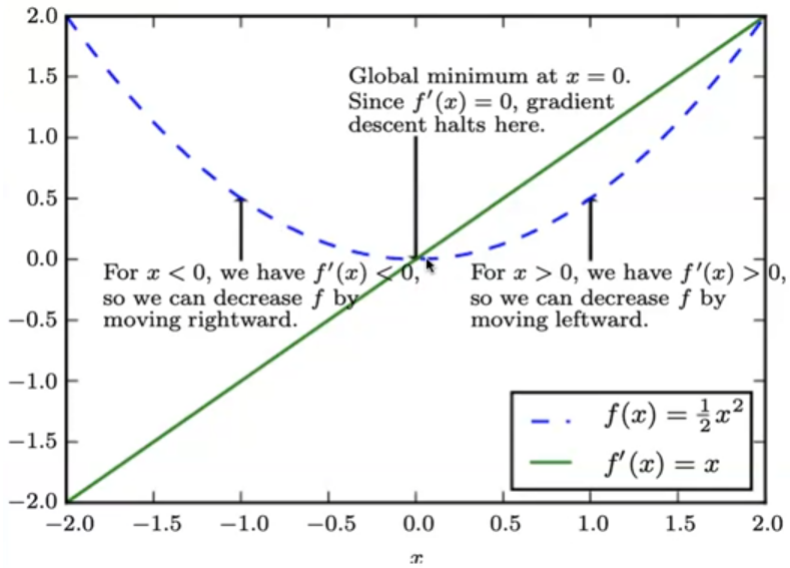
- אם $sign\left(f'\left(x\right)\right) < 0$, אם נלך שמאלה $\left(-\right)$ בצעד ε ערך הפונקציה יגדל ולכן אם נרצה למזער את ערך הפונקציה נלך בכיוון ההפוך.
- ובכל מקרה אם נרצה למזער את ערך הפונקציה בנקודה אז נעבוד לפי הנוסחה:

$$f\left(x - \varepsilon \cdot sign\left(f'\left(x\right)\right)\right) < f\left(x\right)$$

. (וזאת פשוט נוסחת GD במקרה החד ממדי).

- כאשר מתקיים $f'\left(x\right) = 0$ פשוט אין לנו מידע לאיזה כיוון להתקדם כדי למזער את ערך הפונקציה (וזה מתחבר יופי עם העניין של גרדיינט אפסי ולמה אנחנו נרצה להמינע ממצבים כאלו).

דוגמה שמתארת יופי את מה שכתוב:



- לפונקציות הפסד לרוב יש ריבוי נקודות מינימום מקומי, דבר שמקשה לדעת האם האלגוריתם התכנס למינימום הטוב ביותר (המינימום הגלובלי). בנוסף, קיימות נקודות אוכף (saddle points), שבהן בסביבת הנקודה יש כיוונים שבהם הערך גדל וכיוונים שבהם הוא קטן – ולכן לא מדובר במינימום או מקסימום וזה מפריע על תהליך האופטמיזציה.

- בפונקציות convex כל נוקדה שהיא מינימום לוקלי היא גם גלובלי. בנוסף לכך, באופן תיאורטי, לא משנה באיזה נקודה אנו נתחיל, בסוף נגיע למינימום גלובלי (בהנחה ובחוחרים צעד למידה אידיאלי). אך בלמידה עמוקה ורשתות עמוקות, לרוב פונקצית הפסד הופכת למסובכת ולא convex ואז מאבדים את ההבטחה הזו.

- נסיק מכל הדיון הזה, שחייב שהפונקציות ההפסד שלנו יהיו גזירות כדי להשתמש בשיטות GD, גם שנקודנת התחלה מסוימת (אתחול מסוים של פרמטרי הרשת) תוביל אותנו לפתרון מסוים ואז בחירה אחרת של נקודת התחלה תוביל לפתרון אחר.

- וזה מעלה את החשש לכך שאנו עלולים להיתקע במינימום לוקלי מאוד גרוע, אך מבחינה מעשית, לא משנה מה היא נקודת ההתחלה, כמעט תמיד נגיע לתוצאות טובות ודומות. (לרוב לא נתכנס למינימום מאוד גרוע).

  * מהמחקרים הסיקו שככל שהרשת עמוקה יותר, אז מספר הנקודות הלוקליות הגרועות הולך וקטן ורוב נקודות המינימום הלוקלי נותנות לנו ערכים קרובים זה לזה.

- עם זאת, באופן כללי, כן נקבל ערכים שונים עובר נקודות התחלה שונות. ולכן אם נרצה לשפר את האלגוריתם שלנו, נבחן מספר איתחולים ונבדוק את הביצועים שלהם ונבחר הטוב ביותר. אבל לרשת עמוקה ומסובכת, עבור איתחול מסוים יכול לקחת לנו המון זמן לאמן את המדול ולכן זאת גישה לא יעילה.

- המטרה המרכזית שלנו היא להביא את פונקצית ההפסד לערך כמה שיותר נמוך ופחות לדאוג לזה שנגיע לנקודת מינימום גלובלי\לוקלי.

- כשיש לנו פונקציה רבת משתנים, אז יש לנו את נגזרת הפונקציה לפי כל משתנה בנפרד או נגזרת חלקית במילים אחרות: $\frac{\partial f(x)}{\partial x_i}$ וזה מתאר איך הפונקציה משתנה כאשר x_i משתנה.

  - אם נאסוף את כל הנגזרות החלקיות ונשים אותם בוקטור אחד, אז קוראים לוקטור הזה גרדיינט.

- הגרדיינט יתן לנו את הכיוון לפי כל המשתנים של הפונקציה. ועכשיו מכיוון שאנחנו ברב ממדי, אז בנקודה מסוימת יש לנו מספר רב של כיוונים להתקדם בו, אז במונחים שלנו אנחנו נרצה לבחור את הכיוון שימזער לנו הכי הרבה את ערך הפונקציה.

- כדי לבחור את הנקודה הבאה (שתניב לנו את הערך ירידה הכי גדול) נסתכל על כל הכיוונים סביב הנקודה הזו ורוצים לבחור את הכיוון שהפונקציה תרד הכי הרבה אז נחשב:

$$\min_{u,u^Tu=1} u^T\nabla_x f(x) = \min_{u,u^Tu=1} \|u\|\|\nabla_x f(x)\|\cos(\theta)$$

(צורה אחרת למכפלה פנימית) כאשר $\nabla_x f(x)$ הגרדיינט בנקודה x, u וקטור כיוון נרצה לזוז בו, $u^Tu=1$ כדי למדוד כיוון ללא תלות בגודל, ואז $u^T\nabla_x f(x)$ הוא קצב השינוי של הפונקציה בכיוון הזה (וכמובן נרצה את זה שממעזר את הפונקציה ולכן מחפשים את ה- $\min$). כעת ידוע ש- $\|u\|=1$ ו- $\|\nabla_x f(x)\|$ לא תלוי ב- u. גם כן $\cos(\theta)\in[-1,1]$ ולכן נרצה לבחור את הערך -1 כדי למזער את הבטוי הנל. הזווית θ מתארת את הזווית בין הגרדיינט ל- u ולכן כדי לקבל ערך -1 אז נרצה לבחר ב- $\theta=\pi$, כלומר הכיוון של u הוא כמו הכיוון הנגדי של הגרדיינט, או במילים אחרות הכיוון של u הוא מינוס כיוון הגרדיינט, כלומר:

$$\boxed{u = -\frac{\nabla_x f(x)}{\|\nabla_x f(x)\|}}$$

ואז כמו שיודעים, ערך הגרדיינט יכוון לעליה של ערך הפונקציה ולכן מינוס הגרדיינט יוריד את ערך הפונקציה (כמו שראינו במקרה החד ממדי ולכן זה הכללה למקרה הסקלרי). ולכל התהליך הזה קוראים Gradient Decent.

- ואז כדי לעדכן את הנקודה x' נשתמש בנסוחה:

$$\boxed{x' = x - \varepsilon \nabla_x f(x)}$$

כאשר ε הוא גודל הצעד ו- $-\nabla_x f(x)$ הכיוון בו נתקדם.

  - שימו לב: כל הניתוח שעשינו עד כה תקף רק כאשר ε קטן. לכן בחירה בצעד למידה גדול מדי עלולה לשבור את הקירוב, ואף לגרום להגדלת ערך הפונקציה במקום הקטנתו. מצד שני, צעד קטן מדי יוביל להתכנסות איטית, ולכן יש לבחור צעד למידה קטן אך לא קטן מדי.

- אם הגענו לנקודה בה הגרדיינט הוא $\nabla_x f(x) \approx 0$ זה אומר שהאיזור שאנו נמאצים בו הוא "שטוח" ואז אין לו מידע איך אפשר למזער את הפונקציה עוד יותר וזה מצב שאנחנו נתקעים בו ולכן נרצה לברוח ממצבים כאלה (כמו שראינו בשיעורים קודמים)

איך לבחור צעד למידה טוב

נתחיל בהדגרה של נגזרת מסדר שני לפונקציה רבת משתנים קוראים לה מטריצת Hessian שמוגדרת בצורה הבאה:

$$H(x)_{i,j} = \frac{\partial^2}{\partial x_i \partial x_j} f(x)$$

במילים פשוטות, באלכסון יהיה לנו את ערכי הפונקציה כאשר גוזרים פעמיים לפי x_i, ובאשר המקומות במטריצה, יהיה לנו כל הזוגות של לגזור לפי x_i ו- x_j כאשר $i \neq j$.
כמו במקרה הסקלרי, שהיינו גוזריים פעמיים את הפונקציה כדי להבין מה קורה בפונקציה סביבת נקודה מסויימת, למשל אם זו נקודת מינימום או מקסימום או פיתול אז גם כן זה המצב אבל ברב ממד.

- הנגזרת השנייה מספקת מידע על העקמומיות של הפונקציה, ולכן מאפשרת להעריך עד כמה צעד מורד הגרדיאנט מתאים – כלומר אם הצעד קטן מדי, גדול מדי, או בכיוון יציב.

נשתמש בפיתוח טיילור מסדר שני סביב $x^{(0)}$ כאשר g הגרדיינט ו- H ה- Hessian:

$$f(x) \approx f\left(x^{(0)}\right) + \left(x - x^{(0)}\right)^T g + \frac{1}{2}\left(x - x^{(0)}\right)^T H \left(x - x^{(0)}\right)$$

כעת לפי נוסחת GD: $x = x^{(0)} - \varepsilon g$, נציב את זה בטור טיילור מסדר שני ונקבל:

$$f\left(x^{(0)} - \varepsilon g\right) \approx f\left(x^{(0)}\right) - \varepsilon g^T g + \frac{1}{2}\varepsilon^2 \left(g^T H g\right)$$

לפני שהתמשנו ב- Hessian היה לנו פשוט:

$$f\left(x^{(0)}-\varepsilon g\right)\approx f\left(x^{(0)}\right)-\varepsilon g^T g$$

ומן הסתם הנסוחה הזו כן מקטינה את ערך הפונקציה כי $g^T g$ תמיד אי-שלילי. ולכן כעת אפשר להשתמש ב- Hessian כדי שיספק לנו מידע על כך אם הצעד הנוכחי שלנו מגדיל או מקטין את הפונקציה.

- אם $g^T Hg<0$, אז לא משנה מהו ε, $\frac{1}{2}\varepsilon^2\left(g^T Hg\right)$ ערך שלילי ולכן ערך הפונקציה כן יקטן.
- אם $g^T Hg>0$ חיובי גדול מאוד, אז עבור צעד ε מסוים, נוכל להגדיל את הפונקציה כי אננחנו פשוט מוספים לה ערך חיובי גדול וכאן אנחנו בבעיה.

כעת כדי למצוא ערך ε אופטמלי כאשר $g^T Hg>0$ וגם הצעד יקטין את הפונקציה פשוט נגזור את $f\left(x^{(0)}-\varepsilon g\right)$ לפי ε (משוואה ריבועית ב- ε) וערך הקיצון שלה נמצא ב- $-\frac{b}{2a}$, כלומר

$$\varepsilon^*=\frac{g^T g}{g^T Hg}$$

ואנו יודעים כי ערך קיצון זה הוא חיובי כי $a>0$ או $g^T Hg>0$ ואז הפונקציה "מחייכת" ונקודת הקיצון שלה היא מינימום. אם נציב את הערך הזה חזרה בפונקציה, נקבל:

$$f\left(x^{(0)}-\varepsilon g\right)\approx f\left(x^{(0)}\right)-\frac{1}{2}\cdot\frac{\left(g^T g\right)^2}{g^T Hg}$$

והבטוי הזה $-\frac{\left(g^T g\right)^2}{g^T Hg}$ כמובן שלילי כי $g^T g$ חיובי וגם $g^T Hg$ חיובי.
מקרה גרוע של ε מתקבל כאשר ε הוא קטן, כאשר:

$$\varepsilon^*=\frac{g^T g}{g^T Hg}=\frac{1}{\lambda_{\max}}$$

ומקבלים את זה כאשר הגרדיינט g הוא וקטור עצמי המתאים לע"ע $\lambda_{\max}$ של H.
כעת נבחן את המקרה בו מתקבל $\nabla_x f\left(x^*\right)=0$ או פשוט $g=0$, אז נקבל קירוב טיילור מסדר שני סביב x^*:

$$f(x)\approx f\left(x^*\right)+\frac{1}{2}\left(x-x^*\right)^T H\left(x-x^*\right)$$

וזה מזכיר את המקרי הסקלרי כאשר היינו מקבלים $f(x)=0$ ורצים לחקור אם זו נקודת מקסימום או מינימום או פיתול ע"י ערך הנגזרת השנייה ובמקרה שלנו ע"י H.
מתקיים $Hu_i=\lambda_i u_i$ ובנוסף מתקיים $u_i^T u_j=\delta_{ij}$ אם $i=j$ אז $\delta_{ij}=1$ אחרת 0 כלומר u הם אורתונורמליים ולכן באמצעות $\{u_i\}$ ניתן לפרוס את כל המרחב. ואז ניתן לייצג כל וקטור באמצעות

קומבנציה לינאירת של $\{u_i\}$, אז:

$$x - x^* = \sum_i \alpha_i u_i$$

כאשר x^* נקודה בה מתקיים $\nabla_x f(x^*) = 0$. נציב ונקבל:

$$f(x) \approx f(x^*) + \frac{1}{2}\sum_i \lambda_i \alpha_i^2$$

כעת אם מתקיים:

- אם כל ה- $\lambda_i > 0$ אז הנקודה x^* היא מינימום לוקלי
- אם כל ה- $\lambda_i < 0$ אז הנקודה x^* היא מקסימום לוקלי
- אם יש סימנים שונים ל- λ_i אז x^* Saddle point.

וזה פשוט מזכיר לנו את המקרה הסקלרי.

- כאשר הייחס בין $\lambda_{\max}$ ל- $\lambda_{\min}$ (condition number) הוא גדול, מתקבלת פונקציה עם עקמומיות לא אחידה: בכיוון אחד העקומה שטוחה ובכיוון אחר תלולה (כמו פרבולה צרה). לכן הגרדיאנטים אינם מאוזנים – פעם קטנים מאוד ופעם גדולים – וצעד למידה קבוע הופך לבעייתי.
  - עקמומיות קטנה (שטוח): הגרדיאנט קטן, ואם גם צעד הלמידה קטן נתקדם לאט מאוד. כאן נרצה צעד גדול יותר כדי "לברוח" מהר מהאזור השטוח.
  - עקמומיות גדולה (תלול): הגרדיאנט גדול והפונקציה משתנה מהר, ולכן צעד גדול יגרום ל־zigzag או דילוג מעל המינימום. כאן נעדיף צעד קטן וזהיר.
- מסקנה: צעד למידה קבוע אינו מתאים כשיש הבדלי עקמומיות. נרצה צעד אדפטיבי שמותאם ליחס בין $\lambda_{\max}$ ל- $\lambda_{\min}$ גדול באזורים שטוחים וקטן באזורים תלולים.
- אינטואיציה ממני למה שכתוב למעלה:
  - עקמומיות גבוהה (כמו פרבולה צרה): הפונקציה משתנה מהר, וצעד גדול עלול לדלג מעל המינימום – נרצה קצב למידה קטן וזהיר.
  - עקמומיות נמוכה (אזור שטוח): הפונקציה משתנה לאט, וצעד קטן כמעט לא יקדם אותנו – נרצה קצב למידה גדול יותר כדי להתקדם מהר.

פתרון אחד לבעיה הזאת הוא Newton Method:

- נקרב את הפונקציה ע"י טור טיילור מסדר שני לפי סביב $x^{(0)}$:

$$f(x) \approx f\left(x^{(0)}\right) + \left(x - x^{(0)}\right)^T g + \frac{1}{2}\left(x - x^{(0)}\right)^T H \left(x - x^{(0)}\right)$$

- אז העדכון האופטמלי יהי ע"י גזירת הפונקציה הנל לפי $x^{(0)}$ וקבלת הערך המינימלי. אחרי שגוזרים מקבלים נוסחת לעדכון:

$$\boxed{x^* = x^{(0)} - H^{-1}\left(x^{(0)}\right)\nabla_x f\left(x^{(0)}\right)}$$

והשיטה הזו מתאימה את צעד הלמידה לפי ה- curvature של ה- Hessian, וכך נשפר את ההתכנסות.

- יתרונות של השיטה הזו: נוכל להתכנס ממש מהר למינימום כשאר הנקודה הקרובה x^* היא נקודת מינימום.

- חסרונות: כדי לחשב את זה, אנחנו צריכים לחשב נגזרות במטריצה בגודל $n \times n$, וגם לחשב את H^{-1} וגם ההנחה היא ש- $H > 0$ וזה מאוד בעייתי מבחינה חישובית וגם אופטמיזציה.

- שיפורים ל- Newton Method: מן הסתם נרצה קירוב רק בסדר ראשון ואז אנחנו נמנעים מהבעיתיות של ה- Hessian.

תוספת ממני :)

- SGD שזה פשוט כמו GD אך במקום לחשב גרדיאנט על כל הדאטה, מעדכנים את הפרמטרים לאחר כל batch שנבחר.

  - במקום לחכות לכל הדאטה כדי לחשב כיוון "מדויק", משתמשים ב־batch שנותן הערכה מהירה אך רועשת של הגרדיאנט. הרעש הזה מאפשר עדכונים תכופים ומהירים, ולעיתים עוזר להימנע מהיתקעות בנקודות אוכף או במינימום חד, ולעיתים אף דוחף את האופטימיזציה לכיוון מינימום שטוח יותר – פתרון יציב שמכליל טוב יותר

- $SGD + Momentum$: מנגנון ה־Momentum מוסיף "זיכרון" לעדכונים קודמים: כל עדכון מושפע גם מהכיוון הקודם ולא רק מהגרדיאנט הנוכחי.

  - האינטואיציה: כמו כדור שמתגלגל במורד – הוא צובר מהירות בכיוון עקבי, מתעלם מתנודות קטנות, ומתקדם מהר יותר בכיוון הנכון. ואז השאיפה היא להתכנס ל־מינימום אמיתי, ולא להיתקע בנקודות שבהן הפונקציה שטוחה (כמו נקודות אוכף או אזורים עם גרדיאנט כמעט אפסי).

- $Adam$ משלב מנגנון של מומנטום עם צעד למידה אדפטיבי לכל פרמטר, כך שכל עדכון מושפע גם מהכיוון המצטבר של הגרדיאנטים הקודמים וגם מגודלם.
  - האינטואיציה: Adam משלב Momentum עם קצב למידה אדפטיבי – הוא מתקדם לפי כיוון גרדיאנטים מצטבר, אבל מקטין צעדים בפרמטרים עם גרדיאנטים גדולים ומגדיל בפרמטרים עם גרדיאנטים קטנים, וכך מתקבלת התכנסות יציבה ומהירה יותר.

נוסחאות:

- SGD:

$$\theta_{t+1} = \theta_t - \alpha \nabla C_t$$

  - θ_t וקטור הפרמטרים של המודל באיטרציה t
  - α צעד הלמידה
  - ∇C_t הגרדיאנט

- $SGD + MOMENTUM$:

$$\begin{aligned} v_t &= \beta v_{t-1} + \nabla C_t \\ \theta_{t+1} &= \theta_t - \alpha v_t \end{aligned}$$

  - v_t: וקטור המהירות (צבירה של גרדיאנטים קודמים)
  - β: פרמטר המומנטום (כמה "זוכרים" את הכיוון הקודם)

- $Adam$:

$$\begin{aligned} v_t &= \beta_1 v_{t-1} + (1-\beta_1)\nabla C_t \\ s_t &= \beta_2 s_{t-1} + (1-\beta_2)(\nabla C_t)^2 \\ \theta_{t+1} &= \theta_t - \alpha \frac{\hat{v}_t}{\sqrt{\hat{s}_t} + \varepsilon} \end{aligned}$$

  - v_t: מומנט ראשון: ממוצע רץ של הגרדיאנטים מומנט ראשון: ממוצע רץ של הגרדיאנטים (אומדן לכיוון התנועה)

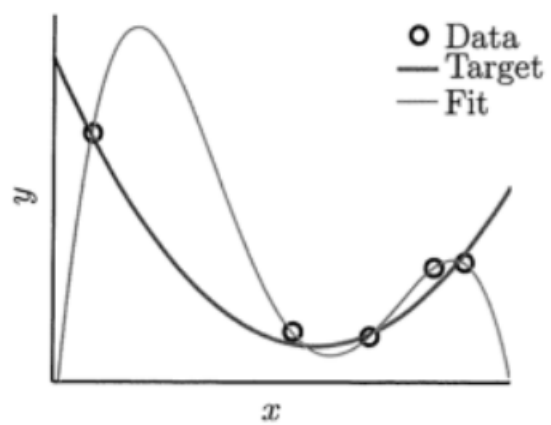
- s_t : מומנט שני: ממוצע רץ של ריבועי הגרדיאנטים (אומדן לעוצמת הגרדיאנט, כלומר סקייל של הצעד).
- β_1 : קובע כמה "זוכרים" גרדיאנטים קודמים (מומנטום)
- β_2 : קובע כמה "זוכרים" את גודל הגרדיאנטים
- $\hat{v}_t = \frac{v_t}{1-\beta_1^t}, \hat{s}_t = \frac{s_t}{1-\beta_2^t}$: תיקון הטיה: מאחר שמתחילים מאפס, הממוצעים בתחילת האימון מוטים כלפי מטה. התיקון מנרמל אותם כך שהצעדים הראשונים לא יהיו קטנים מדי.
- ε : קבוע קטן למניעת חילוק באפס

* בתחילת האימון המומנטים קטנים מדי בגלל אתחול לאפס; תיקון ההטיה מבטל זאת כדי שהעדכונים הראשונים יהיו מדויקים ויעילים.

שיעור 5

מוטיבציה

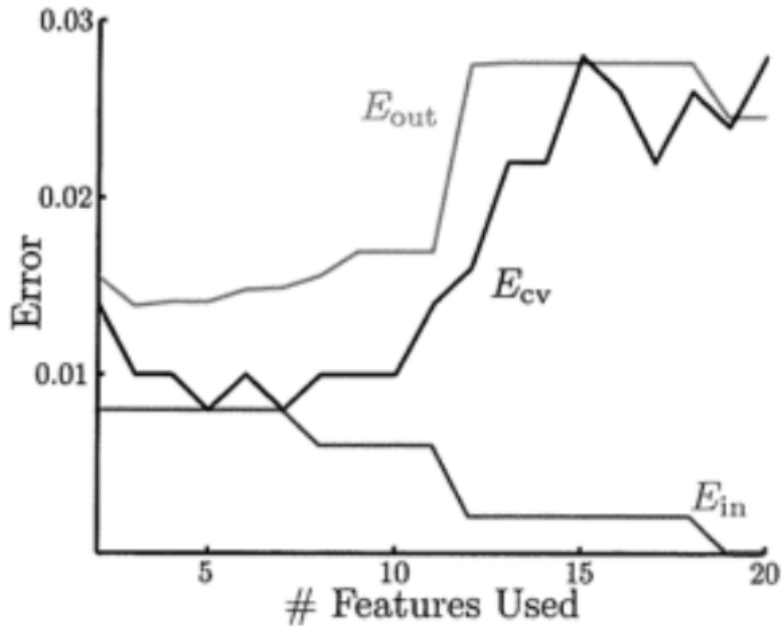
נתבונן באויר הבא:



ישנם שני מודלים: הראשון מורכב יותר ובעל דרגה גבוהה, ולכן מצליח להתאים את עצמו באופן מושלם לנתוני האימון (שגיאה אפסית). לעומתו, המודל השני פשוט יותר ומציג שגיאה קטנה על נתוני האימון. למרות זאת, המודל המורכב מסתמך על מספר דוגמאות מוגבל ונוטה לבצע התאמת יתר – כלומר, הוא לומד גם את הרעש בנתונים. כתוצאה מכך, כאשר נבחן אותו על דוגמאות חדשות מאותה התפלגות, יכולת ההכללה שלו תהיה חלשה יותר.

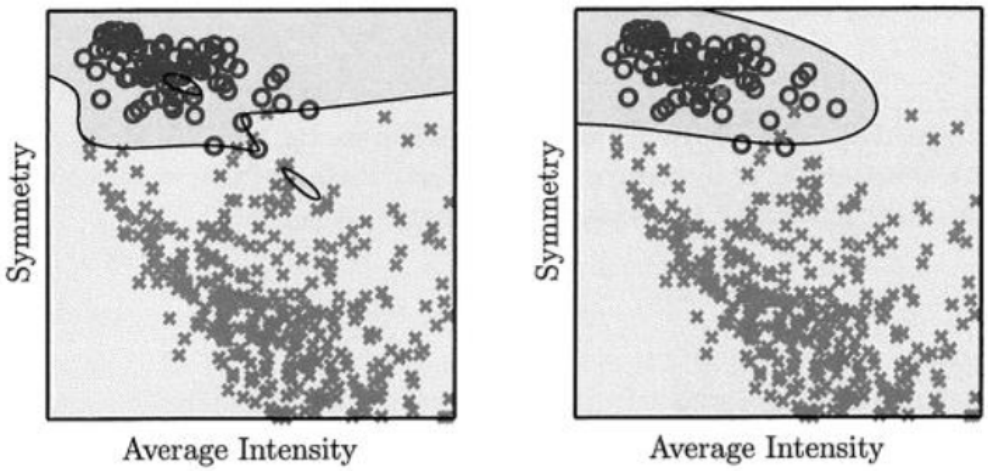
לעומת זאת, המודל הפשוט מצליח ללכוד את המבנה האמיתי של הנתונים בלי להתאים לרעשים, ולכן צפוי להשיג שגיאת הכללה נמוכה יותר.

נתבונן בהאיור הבא:



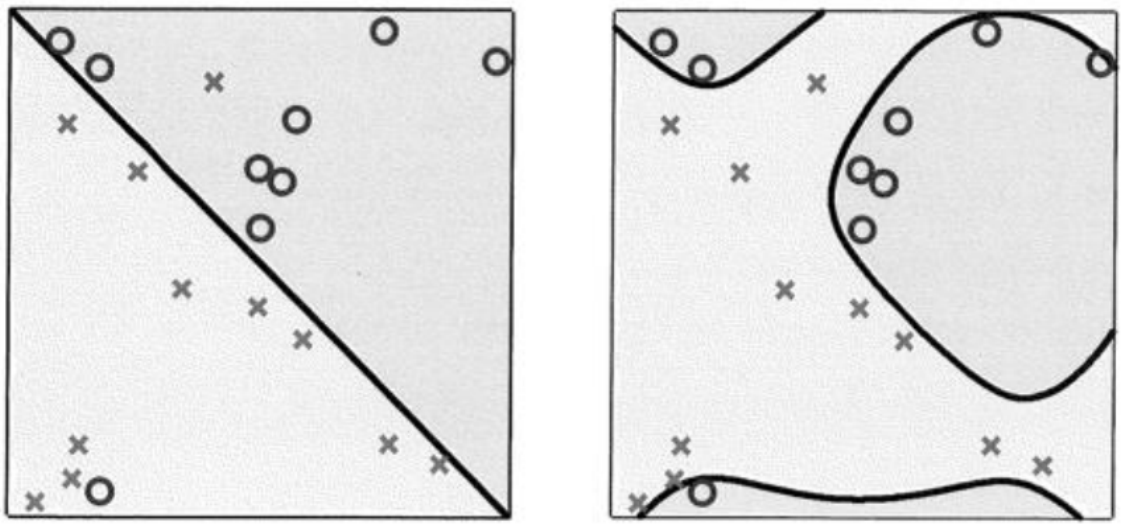
מכאן ניתן להסיק שככל שהמודל מתאים יותר לפיצ'רים, שגיאת האימון E_{in} אמנם קטנה, אך שגיאת ההכללה E_{out} עלולה לגדול. לכן, כאשר עובדים עם מספר גדול של פיצ'רים, המטרה היא לבחור מודל שיאזן בין מזעור E_{in} לבין שמירה על E_{out} נמוך – כלומר, להשיג פשרה טובה בין מורכבות המודל ליכולת ההכללה.

עוד דוגמאות עם מספר פיצרים גדול ומספר פיצרים קטן יותר ויכולת ההכללה:



20-dim classifier (no validation)
$E_{\text{in}} = 0\%$
$E_{\text{out}} = 2.5\%$

6-dim classifier (LOO-CV)
$E_{\text{in}} = 0.8\%$
$E_{\text{out}} = 1.5\%$



(a) Linear fit (b) 4th order polynomial fit

והאויר הזה מצדיק מה שאמרנו למעלה.

רגולריזציה

- בבעיות של ML השאיפה היא לקבל תוצאות טובות על דאטה שלא נצפה ולא רק בסט האימון.
- שיטות רגולריזציה נועדו לשפר את יכולת ההכללה של המודל ע"י מזעור השגיאה על נתונים שלא נצפו בעבר. לעיתים הן אף מעלות מעט את שגיאת האימון, אך זהו טריידאוף רצוי, משום שהמטרה היא להקטין את שגיאת הטסט ולקבל מודל יציב יותר.
- מסקנה: מטרתנו אינה להתאים את המודל באופן מושלם לכל הדוגמאות בסט האימון, אלא לשפר ככל האפשר את יכולת ההכללה שלו – כלומר, את הביצועים על נתונים חדשים שלא נראו במהלך האימון. התאמה מלאה לנתוני האימון עלולה להוביל ל-overfiting. בפרט, ברשתות עמוקות בעלות מספר רב של פרמטרים, קיימת יכולת גבוהה מאוד להתאים את המודל לנתוני האימון, ולכן ללא מנגנונים כמו רגולריזציה קיים סיכון משמעותי להתאמת־יתר.

שיטות רגולריזציה

- אילוץ קשיח: כאשר מספר הפיצ'רים גדול, ניתן להגביל את המודל כך שלא ישתמש בכולם. האילוץ מחייב חלק מהמשקלים להיות אפס, ובכך למעשה מבטל את השפעתם של פיצ'רים מסוימים.
- אילוץ רך: כאשר מספר הפיצ'רים גדול, במקום לאפס משקלים באופן מוחלט, מוסיפים קנס לפונקציית ההפסד על משקלים גדולים. כך המודל עדיין יכול להשתמש בכל הפיצ'רים, אך מעדיף משקלים קטנים יותר (כי אנו מצפים כשיש שינוי קל בקלט גם יהיה שינוי קל בפלט וזה מעודד את הגישה של משקלים קטנים).

- איך בוחרים איזה אילוץ להשתמש בו: לרוב הבחירה נשענת על הידע שלנו על הבעיה ועל הנתונים. אם אנו מעריכים שרבים מהפיצ'רים אינם רלוונטיים, נעדיף אילוץ שמעודד איפוס משקלים, כדי לבצע בחירת פיצ'רים אוטומטית. לעומת זאת, אם רוב הפיצ'רים תורמים למודל, נעדיף אילוץ שמקטין את המשקלים אך לא מאפס אותם. בכל מקרה, לרוב נעדיף מודל פשוט יותר, משום שמודלים פשוטים נוטים לבצע הכללה טובה יותר על נתונים חדשים.

- ככל שהדאטה או הבעיה שלנו מסובכת יותר, אז קל להבין שמהודל שאנחנו בונים או הפונקציה שאנחנו מקרבים $f(x) \approx y$ היא רחוקה מהאמת. לכן לא ממש יעניין אותנו למצוא את המודל שממש פותר את הבעיה (כי שוב לפעמים זה ממש לא אפשרי), אבל, באופן מעשי כאשר מודל יש לו מלא דרגות חופש ומשתמשים בשיטות של רגלוריזציה אז המודל שלנו נהיה גם פחות מורכב וגם בעל יכולת הכללה יותר טובה.

  - אז מודל מורכב עם רגולריזציה יכול להיות עדיף על מודל פשוט – כאשר הבעיה עצמה מורכבת.

- פתרון נאיבי לבעיה שלנו זה להגדיל את סט האימון, ואז מן הסתם תהיה לנו יכולת הכללה טובה יותר כי ראינו מגוון של דוגמאות, אבל דרישה כזאת היא לא פשוטה בכלל כי לא תמיד פשוט להשיג דאטה וגם לא דאטה מתויג.

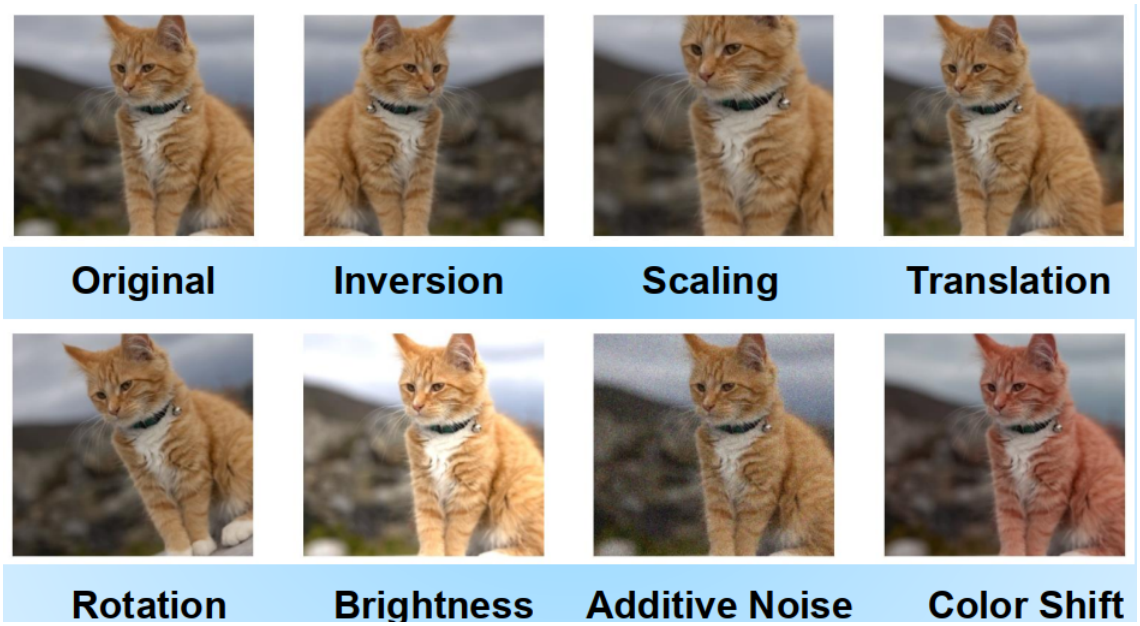
שיטת data augmentation

- ניתן לייצר דאטה סינתטי מתוך הדוגמאות הקיימות. וקוראים לזה data augmentation

  - למשל, ניתן לקחת את הדוגמאות הקיימות ולבצע עליהן טרנספורמציות שונות. בתחום התמונות זה יעיל במיוחד, משום שניתן ליצור מגוון דוגמאות חדשות מתמונה אחת בלי לפגוע במשמעות שלה. טרנספורמציות נפוצות לתמונות כוללות רוטציה, היפוך, זום, חיתוך, שינוי בהירות או צבעים, ואף הוספת רעש קל – וכך מגדילים את הדאטה ומשפרים את יכולת ההכללה של המודל.

    * אינטואיציה: כאשר אנו מזהים אובייקטים בתמונות, שינוי זווית הצילום אמנם יוצר תמונה מעט שונה – אך האובייקט עצמו נשאר זהה. לכן, אם נאמן את המודל גם על גרסאות שעברו טרנספורמציות, הוא ילמד להתמקד במאפיינים החשובים של האובייקט ולא בפרטים שוליים כמו זווית או תאורה. מסיבה זו זוהי טכניקה יעילה ונפוצה בבעיות ראייה ממוחשבת, המשפרת את יכולת ההכללה של המודל.

דוגמה שממחישה את האומר למעלה:



אמנם התמונות "שונות" אבל האובייקט אותו אובייקט בתמונה.

- ברשתות CNN: CNN יודע לזהות שזה אותו אובייקט גם אם הוא זז מעט בתמונה, בעיקר בזכות stride ו- pooling.
  - ה- Stride: גורם לפילטר "לסרוק" את התמונה בקפיצות קטנות, כך שגם אם האובייקט זז קצת – הפילטר עדיין יפגוש את אותם מאפיינים
  - ה- Pooling: מסכם אזור קטן (למשל ע"י max) ולכן המיקום המדויק של המאפיין פחות חשוב – מספיק שהוא קיים באזור.
  - כך הרשת לומדת להיות פחות רגישה להזזות קטנות ולהתמקד בנוכחות המאפיינים, ולכן תזהה שמדובר באותו אובייקט.
- לכן שילוב של רשת CNN + augmentation משפר את יכולת ההכללה, משום שהמודל נחשף לאובייקטים במגוון מצבים ולומד להתמקד במאפיינים החשובים במקום בשינויים קטנים.
- הערה קטנה, שיש לנו ידע מקדים על הדאטה ומה אנחנו מנסיה לפתור, אז כמובן עושים כל דבר בזהירות, למשל בדוגמה למטה רואים שאם נעשה טרנספורמציה מסויימת על הדאטה שלא מתאימה לבעיה שלנו, אז אנחנו עלולים להטעות את המודל כלומר נהרוס את התיוג ואנחנו נרצה להימנע מזה.

- כמובן ששיטה זו עובדת גם בבעיות אחרות, כמו Speech Recognition: ניתן לשפר ביצועים ע"י יצירת וריאציות להקלטות, למשל הוספת רעש רקע, שינוי מהירות הדיבור או תנאי ההקלטה. עם זאת, חשוב שהטרנספורמציות לא יפגעו בתיוג – וזה נקבע לפי מטרת הבעיה. אם המטרה היא לזהות את התוכן הנאמר, זהות הדובר פחות חשובה; אך אם רוצים לזהות מי מדבר, טרנספורמציות שמשנות את הקול עלולות להזיק. לכן יש לבחור טרנספורמציות שמתאימות להגדרת המשימה.

- אז כמו שראינו לשיטה הזו יש חסרות, כלומר הגדלת נתונים לא תמיד עובדת, משום שטרנספורמציות מסוימות עלולות לשנות את משמעות הדאטה או ליצור דוגמאות לא מציאותיות, ובכך לפגוע ביכולת ההכללה של המודל.

שיטת $L^2\ Regularization - Weight\ Decay$

- מטויבציה: שיטה זו מיישמת גישת אילוץ רדך, אנו מעודדים משקלים קטנים באמצעות הוספת איבר ענישה $\|w\|^2$ לפונקציית המטרה שאותה אנו ממזערים. הרעיון הוא ששינוי קטן בקלט יוביל לשינוי קטן בפלט. כאשר המשקלים גדולים, גם שינוי זעיר בקלט עלול לגרום לשינוי חד בפלט, ולכן אנו שואפים להימנע ממצב זה כדי לקבל מודל יציב עם יכולת הכללה טובה יותר.

- לרוב לא מבצעים רגולריזציה על ה-bias משום שהוא כמעט לא מוסיף מורכבות למודל ואינו גורם מרכזי ל-overfiting, אלא רק מזיז את הפלט באופן קבוע ולכן נתמקד רק במשקלות.

- בצורה מתמטית ניתן לכתוב את בעיה שלנו כ:

$$\tilde{J}(w, X, y) = \frac{\alpha}{2} w^T w + J(w, X, y)$$

  כאשר α מקדם הרגולריזציה, נשים לב אם:

  - $\alpha \to 0$ אז נחזור לבעיה המקורית, (נקבל רגרסיה ליניארית רגילה)
  - כאשר $\alpha \to \infty$ אז כדי למזער את הפונקציה, חייבים לבחור את המשקלות להיות אפס או קרובים לזה, כלומר נאפס כל המשקולות.
  - מכאן נסיק שמקדם הרגולריזציה שולט בפשרה בין מורכבות המודל ליכולת ההכללה – ניתן לחשוב עליו כעל כוח שמושך את המשקלים לכיוון 0; ככל שהוא גדול יותר המשקלים יהיו קרובים יותר לאפס (סיכון ל־underfitting), וככל שהוא קטן יותר המודל יהיה גמיש יותר (סיכון ל־overfitting).

- אם גוזרים $\tilde{J}(w, X, y)$ לפי w נקבל:

$$\nabla_w \tilde{J}(w, X, y) = \alpha w + \nabla_w J(w, X, y)$$

ואם נסתכל על נוסחת ה- GD:

$$w \leftarrow w - \varepsilon \left(\alpha w + \nabla_w J\left(w, X, y\right)\right)$$
$$w \leftarrow \left(1 - \varepsilon\alpha\right) w - \varepsilon \left(\nabla_w J\left(w, X, y\right)\right)$$

לרוב בוחרים קטן $0 < \varepsilon < 1$ וגם כן נדאג ש- $\varepsilon\alpha < 1$, כלומר נדאג ש- $|1 - \varepsilon\alpha| < 1$ ואז רואים שאנחנו בכל צעדים מקטנים את ערכו של w ורק אז מוסיפים את העדכון של הגרדיאנט כרגיל. ומכאן הגיע השם של Weight Decay בכל איטרציה המשקלים "דועכים" ומתקרבים ל־0.

- כעת נרצה לבחון את איבר הרגולריזציה ע"י קירוב מסדר 2 של $J\left(w\right)$ סביב הנקודה w^* שהיא מינימום לוקלי, בנקודה הזו הגרדיאנט מתאפס אז נכתוב (כעת עוסקים במקרה שאין איבר רגולריזציה):

$$J\left(w\right) = J\left(w^*\right) + \frac{1}{2}\left(w - w^*\right)^T H \left(w - w^*\right)$$

ומכיוון ש- w^* מינימום לוקלי, אז H $p.s.d$ או $H \succeq 0$. נגזור את $J\left(w\right)$ לפי w ונקבל:

$$\nabla_w J\left(w\right) = H\left(w - w^*\right) = 0$$

ואז $w = w^*$ פתרון אפשרי, אך יתכן עוד פתרונות כי $H \succeq 0$.

- עתה נוסיף את איבר הרגולריזציה ונחזור על התהליך ונקבל:

$$\begin{aligned}\tilde{J}\left(w\right) &= \frac{\alpha}{2} w^T w + J\left(w\right)\\ \nabla_w \tilde{J}\left(w\right) &= \nabla_w \frac{\alpha}{2} w^T w + \nabla_w J\left(w\right)\\ &= \alpha w + H\left(w - w^*\right)\end{aligned}$$

אם נשווה את מה שקיבלנו ל- 0 נקבל:

$$\alpha w + H\left(w - w^*\right) = 0 \Leftrightarrow \left(H + \alpha I\right) w = H w^* \Leftrightarrow \hat{w} = \left(H + \alpha I\right)^{-1} H w^*$$

זה ממש מזכיר את הפיתוח של רגרסיה ליניארית ללא המקדם α (למי שעשה למידה חישובית, וגם כן ראינו את הפיתוח הזה שם ב- Ridge Regression). $\left(H + \alpha I\right)$ הפיכה כי $H \succeq 0$ וגם מדובר כאן ב- $\alpha > 0$, לכן נקבל ש- $H + \alpha I \succ 0$ (כי כל הע"ע של H יהיו גדולים מ- 0 אחרי ההוספה של αI ואז היא הפיכה.. אלגברה ליניארית !ftw) ואגב, זו עוד דרך איך להתמודד עם רגרסיה לינאירת שאין פתרון סגור (שיש אין סוף פתרונות ונרצה לקבל פתרון אחד בלבד.. שכחו ממה שאמרתי כרגע.. טראומה שלי מלמידה חישובית)

- מ- $(H+\alpha I)\,w = Hw^*$ רואים כאשר $\alpha \to 0$ אז $w \to w^*$ ראינו את זה למעלה במקרה ללא α.

כעת נדון במקרה כאשר α גדל: מכיוון ש- H סימטרית וממשית אז קיים לה פירוק ספקטרלי, כלומר:

$$H = Q\Lambda Q^T$$

כאשר Q – מטריצה אורתוגונלית של וקטורים עצמיים.
Λ – מטריצה אלכסונית המכילה את הערכים העצמיים.
נציב את H בנוסחה $\hat{w} = (H+\alpha I)^{-1} Hw^*$ ונקבל:

$$\hat{w} = \left(Q\Lambda Q^T + \alpha QQ^T\right)^{-1} Q\Lambda Q^T w^*$$

מתקיים $QQ^T = I,\ Q^{-1} = Q^T$ ואחרי קצת אלגברה ליניארית נקבל:

$$\hat{w} = Q\left(\Lambda + \alpha I\right)^{-1} \Lambda Q^T w^*$$

נכפיל ב- Q^T ונקבל:

$$\left(Q^T \hat{w}\right) = \left(\Lambda + \alpha I\right)^{-1} \Lambda \left(Q^T w^*\right)$$

לכל ו"ע q_i ניתן ליצג אותו בצורה הבא:

$$\left(q_i^T \hat{w}\right) = \frac{\lambda_i}{\lambda_i + \alpha}\left(q_i^T w^*\right)$$

כאשר $\left(\Lambda + \alpha I\right)_i^{-1} = \frac{1}{\lambda_i + \alpha}$ ו- $\Lambda_i = \lambda_i$. מה ש- α עושה כאן הוא ריסקייל לערכים העצמיים.
אז עבור α קבוע חיובי:

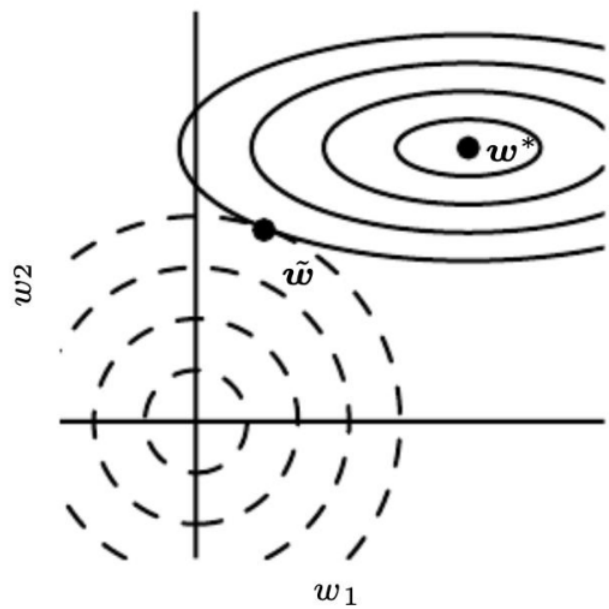
- אם $\lambda_i \ll \alpha$ אז כאילו ויש לנו ויש לנו $\frac{1}{\alpha}$ וזה פשוט מקטין את הע"ע כמעט ל- 0.
- אם $\lambda_i \gg \alpha$ אז כאילו ויש לנו ויש לנו אותו ע"ע ללא שינוי, או במילים אחרות שינוי\ריסקייל מזערי.

- מכאן רואים כי עבור ערכים עצמיים גדולים הרגולריזציה כמעט ואינה משפיעה, בעוד שעבור ערכים עצמיים קטנים היא מכווצת אותם עוד יותר לכיוון האפס.

* ערך עצמי גדול פירושו שהפונקציה כבר מתעקלת חזק בכיוון הזה – כל שינוי קטן במשקל מגדיל מאוד את ה-loss, ולכן האופטימיזציה ממילא שומרת על משקלים קטנים יחסית.
לכן לרגולריזציה כמעט אין מה "לתקן".

* לעומת זאת, ערך עצמי קטן מצביע על אזור שטוח: אפשר לשנות את המשקלים הרבה בלי שה-loss ישתנה כמעט. זה מסוכן כי המשקלים יכולים לגדול ולהוביל לפתרון לא יציב או ל-overfiting. הרגולריזציה מוסיפה עקמומיות מלאכותית, "מהדקת" את הכיוון הזה ומכווצת את המשקלים.

מהאויר למטה ניתן לחשוב על זה כעל "משיכת חבל" בין שני כוחות – פונקציית ה- loss שואפת להגיע ל־w^* כדי להתאים בצורה מיטבית לדאטה , בעוד שהרגולריזציה מושכת את המשקלים לכיוון האפס; הפתרון $\tilde{w}$ הוא הפשרה ביניהם, ולכן הוא מעט פחות מותאם לדאטה אך יציב יותר ופחות נוטה ל־overfiting.



בנוסף, התנועה גדולה יותר בציר x משום שזה הכיוון שבו העקמומיות קטנה יותר (ערך עצמי קטן). כלומר, פונקציית ה-loss כמעט לא "מענישה" שינוי במשקלים בכיוון הזה, ולכן לרגולריזציה קל יותר למשוך את הפתרון לכיוון האפס. לעומת זאת, בציר עם עקמומיות גבוהה y שינוי קטן כבר מגדיל מאוד את ה-loss, ולכן הפתרון כמעט שלא זז שם ביחס לכיוון השני.

שיטת L^1 *Regularization*

- אותו הדבר כמו ב- L^2, רק שההבדל הוא שמוסיפים איבר ענישה $\|w\|_1 = \sum_i |w_i|$ לפונקצית המטרה במקום $\|w\|^2 = \sum_i w_i^2$.
- ואז נקבל את הפונקציה:

$$\tilde{J}(w, X, y) = \alpha\|w\| + J(w, X, y)$$

פונקציה לא גזירה ב- $w = 0$ (כי יש ערך מוחלט) ואז נגזור כאשר $w \neq 0$ ונקבל:

$$\nabla_w \tilde{J}(w, X, y) = \alpha sign(w) + \nabla_w J(w, X, y)$$

ובגלל ה- $sign\left(w\right)$ לא ניתן למצוא פתרון סגור כמו המקרה של L^2.

- נבצע קירוב מסדר ראשון כמו קודם סביב נקודת המינימום w^* ונקבל:

$$\hat{J}\left(w\right)=J\left(w^*\right)+\frac{1}{2}\left(w-w^*\right)^T H\left(w-w^*\right)$$

בכלל שהבעיה מורכבת יותר, נניח ש- H אלכסונית וגם $H\succ 0$. ואז נוכל להתייחס לפתרון הבעיה הזאת לפי איבר איבר, נקבל:

$$J\left(w^*\right)+\sum_i\frac{1}{2}H_{i,i}\left(w_i-w_i^*\right)^2+\alpha\left|w_i\right|$$

ואז אם נרצה לפתור את הבעיה, למצוא w_i שיבא אותה למינימום (נגזור לפי w_i ונשווה ל- 0), בגלל ה- $|w_i|$ נחלק למקרים כאשר $w_i>0$ ו- $w_i<0$ ואז נקבל פתרון לשני המקרים:

$$w_i=sign\left(w_i^*\right)\max\left(\left|w_i^*\right|-\frac{\alpha}{H_{i,i}},0\right)$$

  - רואים לפי הנוסחה, כאשר $w_i=0\leftarrow\left|w_i^*\right|<\frac{\alpha}{H_{i,i}}$, כלומר אנחנו ממש מאפסים את w_i ולא רק מקטינים.
  - גם רואים, כאשר $w_i=sign\left(w_i^*\right)\left(\left|w_i^*\right|-\frac{\alpha}{H_{i,i}}\right)\leftarrow\left|w_i^*\right|>\frac{\alpha}{H_{i,i}}$, אז אנחנו לוקחים את הערך של w_i ומפחיתים ממנו ערך מסוים, כלומר מקטינים אותו אבל לא ממש מאפסים.

- אינטואיציה: רגולריזציית L_2 "מכווצת" את כל המשקלים לכיוון האפס אך כמעט אף פעם לא מאפסת אותם, ולכן מתקבל מודל צפוף שבו כל הפיצ'רים עדיין משתתפים אך בעוצמה קטנה יותר. לעומת זאת, L_1 לא רק מקטינה משקלים אלא ממש מאפסת חלק מהם – כלומר מבצעת בחירת פיצ'רים אוטומטית (זורקת פיצרים לא משמעותיים) ויוצרת מודל דליל (sparse) או במילים אחרות מקטינה מימד.

- ההבדל המרכזי: L_1 מבצע בחירת פיצרים ומוביל לפתרון דליל ופשוט יותר עם פרשנות קלה יותר, בעוד L_2 יוצר פתרון צפוף – שומר על כל הפיצ'רים אך מחליק ומייצב את המודל ע"י הקטנת המשקלים.

שיעור 6

גישות נוספות לרגולריזציה - Ensemble

- נאמן מספר מודלים בלתי תלויים, ובזמן חיזוי, ניקח את החיזוי של כל מודל ואז מה נעשה עם זה תלוי בעיה:

- אם הבעיה בעיית classiftion: אז פשוט נבחר בתוצאה הכי נפוצה, למשל אם יש : 2,2,2,1,3,1, אז נבחר בסיווג 2. (majority vote)
- אם הבעיה בעיית רגרסיה : ניקח כל חיזוי ונחשב ממוצע שלהם וזה יהיה החיזוי הסופי.

- אינטואיציה למה השיטה עובדת: במקום לסמוך על מודל אחד שעלול לטעות, נותנים ל"חוכמת ההמון" להחליט – וכך מקבלים מודל יציב ואמין יותר.

- נרצה שהמודלים יהיו בלתי תלויים, כי אינטואיטיבית אם נבקש מאותו אדם להחליט על אותה משימה 10 פעמים – כנראה שיבחר במשימה A כל פעם. לעומת זאת, אם ניקח אנשים עם רקע שונה (בלתי תלויים) שהטעויות שלהם כנראה שונות, ובכל זאת יבחרו במשימה A – נהיה בטוחים בהחלטה יותר מאשר אם נשתמש ב-10 מודלים כמעט זהים.

- הגישה הזו נותנת יכולת הכללה טובה יותר ומסייעת למנוע התאמת יתר, משום ששילוב כמה מודלים מוביל לרוב לדיוק גבוה יותר וליכולת טובה יותר לסנן רעש – זאת בזכות טעויות שונות של המודלים שמאזנות זו את זו.

כעת נכנס לעניינים וניתן הצדקות למה שאמרנו בצורה מתמטית: נניח שיש לנו בעיית רגרסיה ואימנו מספר מודלים למצוא את הקשר בין x ל- y. אז:

$$y_{com}(x) = \frac{1}{M}\sum_{m=1}^{M} y_m(x)$$

יש לנו M מודלים וכל מודל יחזיר לנו חיזוי שונה y_m, ואז $y_{com}(x)$ יהיה ממוצע החיזוי של כל המודלים.
בהינתן $h(x)$ הערך האמיתי של דוגמה x (החיזוי הנכון של הדוגמה) אז נכתוב:

$$y_m(x) = h(x) + \varepsilon_m(x)$$

כאשר $\varepsilon_m(x)$ היא טעות המודל ה- m.
נניח:

$$\begin{aligned} E(\varepsilon_m(x)) &= 0 \\ Var(\varepsilon_m(x)) &= E\left(\varepsilon_m(x)^2\right) = \sigma^2 \\ E(\varepsilon_m(x)\,\varepsilon_n(x)) &= \rho\sigma^2, m \neq n \end{aligned}$$

נכתוב את השגיאה הריבועית הממוצעת למודל m:

$$E\left((y_m(x) - h(x))^2\right) = E\left(\varepsilon_m(x)^2\right) = \sigma^2$$

נעשה אותו הדבר ל y_{com} ונקבל:

$$E\left(\left(y_{com}\left(x\right)-h\left(x\right)\right)^{2}\right)=E\left(\left(\left(\frac{1}{M}\sum_{m=1}^{M}y_{m}\left(x\right)\right)-h\left(x\right)\right)^{2}\right)$$

מכיוון ש- $h\left(x\right)$ לא תלוי ב- m אז נכניס אותו לסכימה ונקבל :

$$E\left(\left(\frac{1}{M}\sum_{m=1}^{M}\left(y_{m}\left(x\right)-h\left(x\right)\right)\right)^{2}\right)=E\left(\left(\frac{1}{M}\sum_{m=1}^{M}\varepsilon_{m}\left(x\right)\right)^{2}\right)$$

נוציא $\frac{1}{M}$ ונרשום אם הסכום בצורה כללית:

$$\begin{aligned}\frac{1}{M^{2}}E\left(\sum_{m=1}^{M}\sum_{n=1}^{M}\varepsilon_{m}\left(x\right)\varepsilon_{n}\left(x\right)\right)&=\frac{1}{M}\sigma^{2}+\frac{M-1}{M}\rho\sigma^{2}\\&=\frac{1+\left(M-1\right)\rho}{M}\sigma^{2}\end{aligned}$$

כאשר $m=n$ נקבל $M\cdot E\left(\varepsilon_{m}\left(x\right)^{2}\right)=M\cdot\sigma^{2}$ וכאשר $m\neq n$ נקבל: $\left(M^{2}-M\right)\cdot E\left(\varepsilon_{m}\left(x\right)\varepsilon_{n}\left(x\right)\right)=\left(M^{2}-M\right)\rho\sigma^{2}$ ואז אם נבפיל את כל זה ב- $\frac{1}{M^{2}}$ נקבל את:

$$\frac{1+\left(M-1\right)\rho}{M}\sigma^{2}$$

- אם $\rho=0$, זה אומר שהמודלים בלתי תלויים ואז השונות תקטן כמספר המודלים – נקבל $\frac{1}{M}\sigma^{2}$
- אם $\rho=1$, זה אומר שהמודלים תלויים (עושים שגיאות דומות) ואז לא נרוויח מהם בכלל, נקבל σ^{2} בדיוק כמו מודל יחיד.
  - זה מתחבר לנו יופי עם האינטואיציה שכתבנו למעלה.
- מסקנה:
  - ככל ששגיאות המודלים פחות מתואמות, הממוצע מפחית את השגיאה ומשפר את הביצועים; אך כאשר השגיאות מתואמות – אין כמעט יתרון לשילוב.
  - ממוצע מודלים תמיד משפר או לפחות לא פוגע בביצועים, וככל ששגיאות המודלים פחות תלויות זו בזו – השיפור יהיה משמעותי יותר.
- הגישה הכללית: נאמן כמה מודלים שעושים טעויות שונות ונשלב ביניהם – כך הטעויות מתקזזות והביצועים משתפרים. את הגיוון ניתן להשיג באמצעות שימוש בסוגי מודלים שונים או באלגוריתמי אימון / פונקציות מטרה שונות.

גישה נוספת ל- Ensemble היא Bagging

- במקום לשנות את המודל או את הפרמטרים, נשתמש באותו מודל בדיוק, אך נאמן כל אחד על דאטה שונה – ולכן גם החיזויים יהיו שונים.
  - למה לא פשוט לחלק את הדאטה ל־ M קבוצות? כי כך כל מודל מאומן על פחות נתונים, הדאטה נעשה דליל יותר, והסיכון ל־overfiting דווקא גדל – בניגוד לרעיון של לנצל כמה שיותר מידע מהדאטה (data augmentation).
  - מה שנעשה הוא לאמן את המודל על דאטה סטים שונים שנבנים מתוך הדאטה סט המקורי, כאשר גודל כל דאטה סט נשאר כמו הגודל המקורי. איך זה ייתכן? נבחר דוגמה עם החזרה – כלומר כל דוגמה יכולה להיבחר כמה פעמים או לא להיבחר בכלל. לכן, גם אם יש לנו 1000 דוגמאות, ההסתברות לקבל שני דאטה סטים זהים היא בערך $\frac{1}{1000^{1000}}$ כלומר זניחה לחלוטין.
  - למרות שגישה זו משפרת לרוב את הביצועים, עדיין קיימת תלות מסוימת בין המודלים – לא רק משום שמדובר באותו מודל, אלא גם כי כל הדאטה סטים נבנים מאותו מקור ולכן יש ביניהם חפיפה. כתוצאה מכך, לא נקבל את האפקט המלא של מודלים בלתי תלויים.
- בכללי בגישת ה- Ensemble:
  - מפחיתה משמעותית את שגיאת ההכללה ולעיתים מוביל לשיפור ניכר בביצועים.
  - חיסרון: דורש יותר חישוב וזיכרון משום שיש לאמן ולשמור מספר מודלים.
  - זהירות בהשוואות: לרוב לא משתמשים ב־Ensemble בבנצ'מרקים מדעיים, כי כמעט כל אלגוריתם יכול להשתפר וכך ההשוואה נעשית פחות הוגנת.
  - בפועל מומלץ להשתמש בממוצע מודלים – אך להימנע ממנו כשמשווים בין אלגוריתמים.

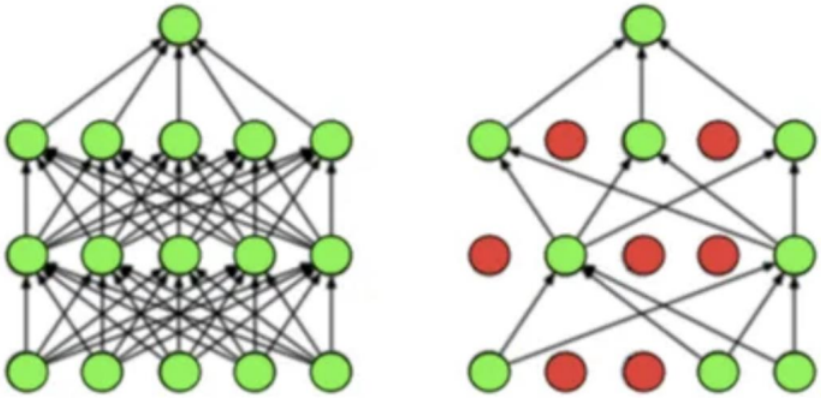
גישת Dropout

- מוטיבציה: רשת עמוקה יכולה ללמוד קשרים מורכבים מאוד בין x ל- y לא משנה כמה מסובך, אך יכולת זו עלולה להוביל לשינון הדאטה (overfiting) מודל מורכב מדי עלול "לתפוס" גם פרטים קטנים ורעש, במיוחד כאשר ניתן להוסיף שכבות ונוירונים כמעט ללא הגבלה בעוד שהדאטה עצמו מוגבל.
- הרעיון המרכזי הוא לקבל את היתרונות של Ensemble בלי העלות הגבוהה של חישוב וזיכרון. במקום לאמן הרבה מודלים שונים, משתמשים באותה רשת – אך בכל איטרציה (כל דוגמה) מכבים אקראית חלק מהנוירונים (מלבד שכבת הפלט). כך, אם ברשת יש n נוירונים, לכל נוירון יש מצב של דלוק או כבוי (0 או 1) לכן קיימות עד 2^n תתי־רשתות אפשריות. בכל פעם שמכבים נוירונים אחרים מתקבלת רשת דלילה ושונה, ואינטואיטיבית זה דומה לאימון מספר עצום של מודלים – אך ללא העלות החישובית והזיכרון הנלווים ל- Ensemble אמיתי וזה משוגע!

- ניתן לבחור הסתברות p לאיפוס נוירונים בכל שכבה. אמפירית נמצא כי כדאי לאפס באקראי כ־20% בשכבת הקלט וכ- 50% בשכבות החבויות אך, בפועל אפשר לכוון את ההסתברויות הללו באמצעות סט ולידציה.

- p נמוך מדי יכול לאפס כמעט כל הרשת ואז נהרוס את היכולת של ביטוי קשרים מסובכים בין הקלט לפלט מה שנקרא underfiting. וההפך נכון, ככל ש- p גבוה, כמעט ולא עשינו דבר ונחזר לבעיה של overfiting.

ציור שממחיש תת-רשת אחרי dropout:



- בזמן האימון כל דוגמה "רואה" תת־רשת אחרת, ולכן תאורטית קיימות 2^n תוצאות, רשתות אפשריות, אך בזמן הטסט מיצוע של כולן אינו מעשי. לכן מאמנים עם Dropout כרגיל, ובזמן הטסט משתמשים ברשת המלאה – כאשר המשקולות מוכפלות ב- p, כלומר $p \cdot w$ (כאשר p הוא הסתברות ההשארה של נוירון פעיל לפי השכבה) וכך מתקבל קירוב לממוצע התחזיות של כל תתי־הרשתות.

  - אינטוציה לזה: קיימת הוכחה מתמטית למקרה של פונקציות הפעלה לינאריות, אך גם כשזה לא המצב ניתן לחשוב על כך שכל נוירון היה פעיל בערך p מהפעמים במהלך האימון. לכן, הכפלת המשקולות ב- p בזמן הטסט מהווה קירוב טוב לממוצע התחזיות של כל תתי־הרשתות.

- הערה: לשכבת הפלט לא עושים Dropout!

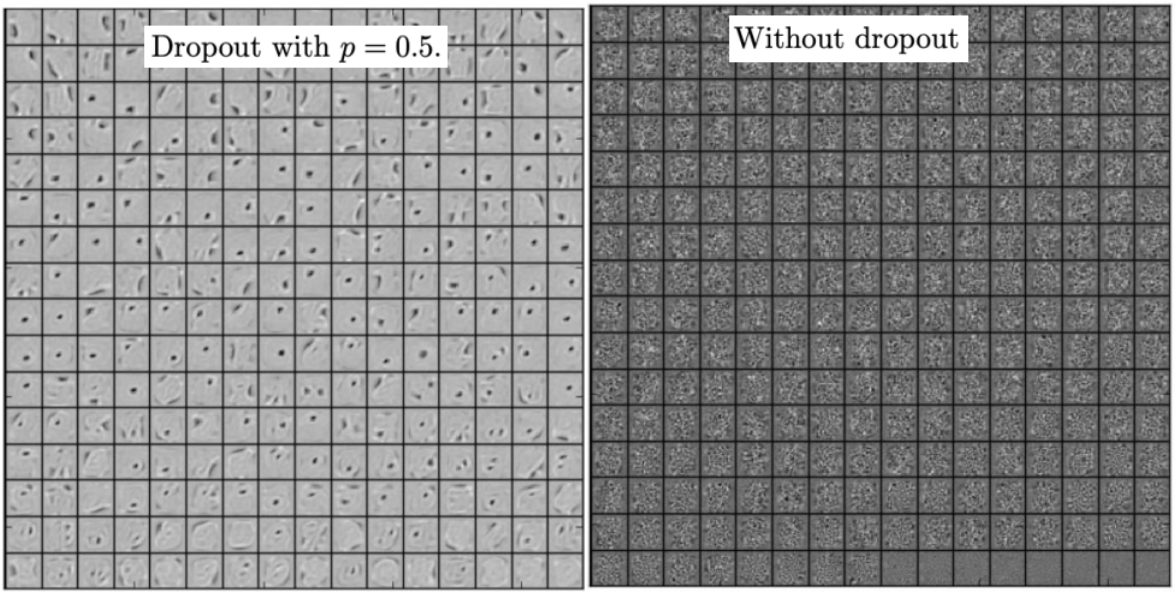
- לרוב השיטה הזו מנצחת כל שיטת רגולריזציה אחרת.

- אימון עם הגישה הזו משולב עם SGD וההבדל הפשוט כמו שבכר אמרנו, לכל דוגמה ב- mini-batch מאמנים אותה על תת-רשת אחרת. בהתאם לכך, גם ה-forward וה- backprop מתבצעים על אותן תתי־רשתות שנדגמו באותה איטרציה.

- כאשר אין Dropout וכל הנוירונים מחוברים, נוצרת תלות ביניהם – נוירונים יכולים להסתמך על אחרים שיתקנו טעויות שלהם. לעומת זאת, עם Dropout כל נוירון נאלץ ללמוד באופן עצמאי יותר, בלי להישען על נוירונים אחרים, וכך מתקבל מודל רובסטי, חזק ויעיל יותר.

  - נניח שמשימה מסוימת מתבצעת דרך מספר שלבים שכל אחד תלוי בקודם. אם שלב אחד חסר – כל המשימה נכשלת, וזה מעיד על תלות גבוהה. לעומת זאת, אם לעיתים שלבים אקראיים "נעלמים" והמערכת עדיין מצליחה להשלים את המשימה, סימן שהיא למדה לפצות ולהסתגל. כך מתקבל מודל חזק ורובסטי יותר, שאינו נשען על רצף קבוע של שלבים.

דוגמה שממחישה תלות ואי־תלות בין נוירונים: בוצע Dropout על שכבה, ובכל ריבוע ניתן לראות את הערכים שנלמדו ע"י כל נוירון. ללא Dropout הנוירונים נראים כמעט זהים – כלומר לומדים מאפיינים דומים ותלויים זה בזה. לעומת זאת, עם Dropout ניתן לראות שכל נוירון למד מאפיינים שונים. בדוגמה מעולם התמונות, המשמעות היא שכל נוירון מזהה תכונה משמעותית אחרת – וכך מתקבלת יכולת הכללה גבוהה יותר.



- ניתן לפעול גם בצורה מעט שונה: במקום להכפיל את המשקולות ב- p בזמן הטסט, מכפילים את המשקולות ב- $\frac{1}{p}$ כבר בזמן האימון. כך בזמן הטסט משתמשים ברשת המקורית ללא שינוי – וגישה זו שקולה לגישה הראשונה.

- הכללה ל-Dropout: במקום להשתמש ב- p כמשתנה ברנולי (0 או 1), ניתן להחליף אותו במשתנה מקרי מהתפלגות $N(1, \sigma^2)$. בוחרים תוחלת 1 כדי שבזמן הטסט נוכל להשתמש ברשת המקורית ללא תיקון, ואת השונות ניתן לקבוע לפי ערך הדומה לשונות של משתנה ברנולי $\left(\frac{1-p}{p}\right)$ או לכוון באמצעות סט ולידציה. גישה זו אף הראתה תוצאות דומות ולעיתים טובות יותר, אך בניגוד ל-Dropout קלאסי היא מוסיפה רעש לרשת במקום לאפס נוירונים – כלומר הנוירונים אינם רק כבויים או פעילים, אלא מקבלים ערכים רציפים.

- חסרנות של Dropout:

  - לרוב הוא מגדיל את זמן האימון פי 2-3, משום שכל דוגמה מאומנת על תת־רשת מעט שונה. כתוצאה מכך עדכוני הפרמטרים נעשים רועשים יותר, ולכן נדרש יותר זמן עד להתכנסות.
  - עם זאת, מדובר בשיטת רגולריזציה יעילה מאוד שמפחיתה התאמת יתר ו-co-adaptation, במחיר של זמן אימון ארוך יותר.

במקרים פשוטים, כגון רגרסיה לינארית, ניתן לקבל את היתרונות של Dropout מבלי להתמודד עם האקראיות של השיטה, באמצעות מיצוע הרעש וגזירת רגולריזציה דטרמיניסטית שקולה. כלומר, בהינתן בעיית רגרסיה לינארית נרצה למזער:

$$\min_w \|y - Xw\|^2$$

אז מבצעים Dropout ע"י הוספת מקדם $R_{i,j}$ לאיברים של X כך ש- $R_{i,j} \sim Bernoulli\,(p)$, כלומר $\tilde{X} = R_{i,j}X_{i,j}$, אז:

$$E_R\left(\tilde{X}_{i,j}\right) = pX_{i,j}$$
$$E_R\left(\tilde{X}_{i,j}\tilde{X}_{k,l}\right) = \begin{cases} p^2 X_{i,j}X_{k,l} & o.w. \\ pX_{i,j}^2 & i=k, j=l \end{cases}$$

אז כעת נרצה למזער את התוחלת של אותה בעיה אבל עם Dropout, כלומר:

$$\min_w E_R\left(\|y - \tilde{X}w\|^2\right)$$

ארשום את התוצאה הסופית, את הפיתוחים ניתן לראות במצגת האחרונה של שי (פשוט פותחים את $\|y - \tilde{X}w\|^2$ ומשתמים בהגדרות של התחולת שכתבנו) ואז נקבל אחרי כל זה:

$$E_R\left(\|y - \tilde{X}w\|^2\right) = \|y - pXw\|^2 + p\,(1-p)\,\|\Gamma w\|^2,$$
$$\Gamma = \left(diag\left(X^T X\right)\right)^{\frac{1}{2}}$$

אז נקבל ש $\min_w E_R\left(\|y - \tilde{X}w\|^2\right)$ שקול בעצם ל-

$$\min_w \|y - pXw\|^2 + p\,(1-p)\,\|\Gamma w\|^2$$

שזה ממש מזכיר רגולריזציה של L_2.

אינטואיציה: במקום לאמן הרבה מודלים עם רעש אקראי, אנחנו ממצעים את השפעת הרעש מראש – והתוצאה היא איבר רגולריזציה שמעניש משקולות גדולות. בפרט, מאחר שהענישה תלויה ב- $X^T X$,

פיצ'רים עם שונות גבוהה גוררים קנס גדול יותר כאשר המשקל שלהם גדול, ולכן המודל נוטה להקטין משקולות ולבחור פתרון יציב יותר. כלומר, Dropout מתנהג כאן כמו Ridge: הוא מקטין את המשקולות, מפחית תלות בפיצ'רים בודדים ומשפר הכללה בלי צורך באקראיות בפועל.