# (20942) מבוא ללמידה חישובית | סיכום הרצאה 11

מנחה: ד"ר שי מימון
סמסטר: 2022א'
נכתב על ידי: מתן כהן

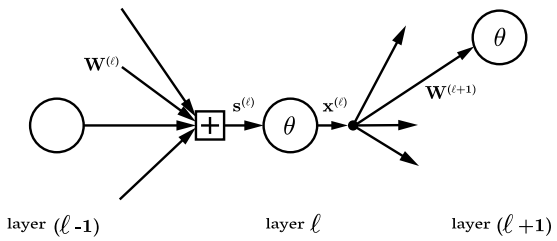
## 1 Backpropagation

מהות העניין הוא חישוב יעיל של Gradient Descent על פני כל שכבות הרשת.
ניזכר כי על מנת לחשב את הגרדיאנט של $E_{in}$ יש צורך בנגזרת שלו ביחס למטריצת המשקלים

$$\frac{\partial E_{in}}{\partial W^{(\ell)}} = \frac{1}{N}\sum_{n=1}^{N}\frac{\partial e_n}{\partial W^{(\ell)}}$$

נסמן:

- $e$ - שגיאה על נקודה מסוימת $(\mathbf{x}_n, y_n)$
- $w_{ij}^{(\ell)}$ - משקל הקשת בין נוירון $i$ משכבה $\ell-1$ לנוירון $j$ לשכבה $\ell$




תחילה ניזכר כי:

$$\mathbf{s}_j^{\ell)} = \sum_{\alpha=0}^{d^{(\ell-1)}} w_{\alpha j}^{(\ell)}\mathbf{x}_\alpha^{(\ell-1)} \qquad \Rightarrow \frac{\partial \mathbf{s}_j^{(\ell)}}{\partial w_{ij}^{(\ell)}} = \mathbf{x}_i^{(\ell-1)} \tag{1.1}$$

כעת נבחין כי המקום בו המשקולת $w_{ij}^{(\ell)}$ משפיעה על משתני הכניסה לשכבה $\ell$ הוא $\mathbf{s}_j^{(\ell)}$ ולכן:

$$\frac{\partial e}{\partial w_{ij}^{(\ell)}} \overset{\text{chain rule}}{=} \frac{\partial \mathbf{s}_j^{(\ell)}}{\partial w_{ij}^{(\ell)}} \cdot \frac{\partial e}{\partial \mathbf{s}_j^{(\ell)}} \overset{1.1}{=} \mathbf{x}_i^{(\ell-1)} \cdot \frac{\partial e}{\partial \mathbf{s}_j^{(\ell)}}$$

נסמן את ה- sensitivity:

$$\frac{\partial e}{\partial \mathbf{s}_j^{(\ell)}} = \boldsymbol{\delta}_j^{(\ell)}$$

ונקבל:

$$\frac{\partial e}{\partial w_{ij}^{(\ell)}} = \mathbf{x}_i^{(\ell-1)} \cdot \boldsymbol{\delta}_j^{(\ell)}$$

בעזרת מה שהראינו נוכל לכתוב:

$$\frac{\partial e}{\partial W^{(\ell)}} = \underbrace{\mathbf{x}^{(\ell-1)}}_{\left(d^{(\ell-1)}+1\right)\times 1} \cdot \left(\underbrace{\boldsymbol{\delta}^{(\ell)}}_{d^{(\ell)}\times 1}\right)^T$$

כאשר הממד של התוצאה הוא

$$\left(d^{(\ell-1)}+1\right)\times d^{(\ell)}$$

כעת נתמודד עם $\boldsymbol{\delta}_j^{(\ell)}$, לשם כך נבחין כי נוכל להשתמש ב $\mathbf{x}^{(\ell)}$ וכיוון שכל אלמנט $\mathbf{s}_j^{(\ell)}$ משפיע רק על אלמנט $\mathbf{x}_j^{(\ell)}$ נוכל להשתמש בזה על מנת לכתוב:

$$\boldsymbol{\delta}_j^{(\ell)} = \frac{\partial e}{\partial \mathbf{s}_j^{(\ell)}} = \frac{\partial e}{\partial \mathbf{x}_j^{(\ell)}} \cdot \frac{\partial \mathbf{x}_j^{(\ell)}}{\partial \mathbf{s}_j^{(\ell)}} = \frac{\partial e}{\partial \mathbf{x}_j^{(\ell)}} \cdot \theta'\left(\mathbf{s}_j^{(\ell)}\right)$$

כעת, אם נוכל לייצר מנגנון נוח לחישוב $\delta$ נוכל להתמודד איתה בכל שכבה בעת ה- Backpropagation בצורה פשוטה.
לשם כך נתבונן בביטוי $\frac{\partial e}{\partial \mathbf{x}_j^{(\ell)}}$ בו השתמשנו על מנת לבטא את $\boldsymbol{\delta}_j^{(\ell)}$ וזאת כאשר נזכור שכל $\mathbf{x}_j^{(\ell)}$ משפיע על **כל** $\mathbf{s}_k^{(\ell+1)}$ מ**השכבה הבאה**:

$$\frac{\partial e}{\partial \mathbf{x}_j^{(\ell)}} = \sum_{k=1}^{d^{(\ell+1)}} \frac{\partial \mathbf{s}_k^{(\ell+1)}}{\partial \mathbf{x}_j^{(\ell)}} \cdot \frac{\partial e}{\partial \mathbf{s}_k^{(\ell+1)}} = \sum_{k=1}^{d^{(\ell+1)}} w_{jk}^{(\ell+1)} \boldsymbol{\delta}_k^{(\ell+1)}$$

קיבלנו בסוף **שלכל** $j = 1, \ldots, d^{(\ell)}$:

$$\boldsymbol{\delta}_j^{(\ell)} = \theta'\left(\mathbf{s}_j^{(\ell)}\right) \sum_{k=1}^{d^{(\ell+1)}} w_{jk}^{(\ell+1)} \boldsymbol{\delta}_k^{(\ell+1)}$$

נגדיר שהתסכלות על איבר $i$ עד איבר $j$ בוקטור $\mathbf{v}$ יכתב:

$$[\mathbf{v}]_i^j$$

מכפלת איבר באיבר של וקטורים $\mathbf{v}, \mathbf{u}$ תכתב:

$$\mathbf{v} \otimes \mathbf{u}$$

ועל פי הסימון נוכל לכתוב את $\boldsymbol{\delta}^{(\ell)}$:

$$\boxed{\boldsymbol{\delta}^{(\ell)} = \theta'\left(\mathbf{s}^{(\ell)}\right) \otimes \left[W^{(\ell+1)} \boldsymbol{\delta}^{(\ell+1)}\right]_1^{d^{(\ell)}}}$$

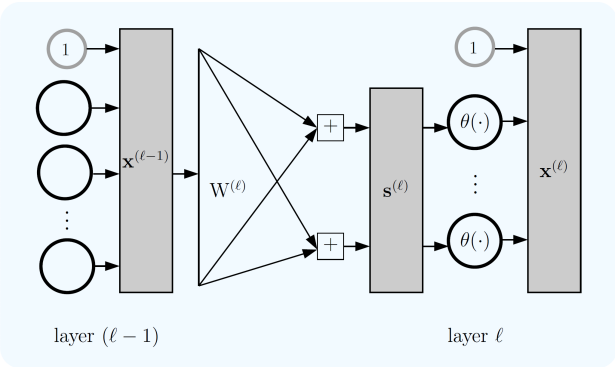
כעת נוכל להסתכל על כלל התהליך.

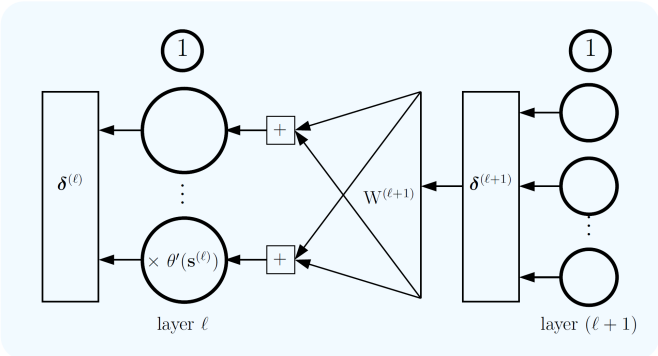
## 1.1 תהליך ה- Backpropagation

ניזכר כי בהתחלה עשינו forward propagation על מנת לחשב את האיקסים $\mathbf{x}^{(\ell)}$ משכבת ה-input ועד שכבת ה-output כאשר **חישבנו** את $\mathbf{x}^{(\ell)}$ בעזרת $\mathbf{x}^{(\ell-1)}$.

ב-backpropagation נרצה לחשב את הדלתאות $\boldsymbol{\delta}^{(\ell)}$ משכבת ה-output לאחור עד שכבת ה-input כאשר **נחשב** את $\boldsymbol{\delta}^{(\ell)}$ בעזרת $\boldsymbol{\delta}^{(\ell+1)}$.

ישנו דמיון בין שני התהליכים!




### 1.1.1 התחלת תהליך ה-backpropagtaion

- חשוב לזכור שבשכבה האחרונה אנחנו מדברים על סקלרים

נרצה להתחיל עם $\boldsymbol{\delta}^{(L)}$ (הרי $L$ זו השכבה האחרונה) ולכן על מנת לחשב את הדלתא:

$$\begin{aligned}
\boldsymbol{\delta}^{(L)} &= \frac{\partial e}{\partial \mathbf{s}^{(L)}} \overset{\text{last layer}}{=} \frac{\partial e}{\partial s^{(L)}} \\
&= \frac{\partial}{\partial s^{(L)}} \left(x^{(L)} - y\right)^2 \\
&= 2\left(x^{(L)} - y\right) \cdot \frac{\partial x^{(L)}}{\partial s^{(L)}} \\
&= 2\left(x^{(L)} - y\right) \cdot \theta'\left(s^{(L)}\right)
\end{aligned}$$

כעת, **אם** $\theta$ היא **tanh** הרי ש- $\theta'\left(s^{(L)}\right) = 1 - \left(x^{(L)}\right)^2$ ולכן:

$$\boldsymbol{\delta}^{(L)} = 2\left(x^{(L)} - y\right) \cdot \left(1 - \left(x^{(L)}\right)^2\right)$$

**אם** מדובר ברגרסיה והשתמשנו לדוגמה בפונקצית הזהות הרי ש- $\theta'\left(s^{(L)}\right) = 1$:

$$\boldsymbol{\delta}^{(L)} = 2\left(x^{(L)} - y\right)$$

### 1.1.2 אלגוריתם ה-Backpropagation

**אלגוריתם 1** Backpropagation to compute sensitivities $\delta^{(\ell)}$

**Input: a data point** $(\mathbf{x}, y)$

(1) Run forward propogation on $\mathbf{x}$ to compute and save:

$$\mathbf{s}^{(\ell)} \quad \text{for } \ell = 1, \ldots, L;$$
$$\mathbf{x}^{(\ell)} \quad \text{for } \ell = 0, \ldots, L.$$

(2) $\delta^{(L)} \leftarrow 2\left(x^{(L)} - y\right)\theta'\left(s^{(L)}\right)$ **[Initialization]**

$$\theta'\left(s^{(L)}\right) = \begin{cases} 1 - \left(x^{(L)}\right)^2 & \theta(s) = \tanh(s); \\ 1 & \theta(s) = s \end{cases}$$

(3) **for** $\ell = L - 1$ to $1$ **do** **[Back-Propagation]**

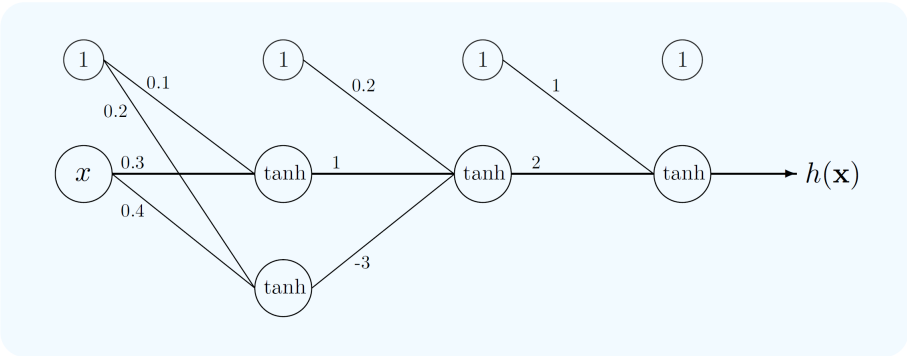
(a) Let $\theta'\left(\mathbf{s}^{(\ell)}\right) = \left[1 - \mathbf{x}^{(\ell)} \otimes \mathbf{x}^{(\ell)}\right]_1^{d^{(\ell)}}$ $\quad \theta = \tanh$

(b) Compute the sensitivity $\boldsymbol{\delta}^{(\ell)}$ from $\boldsymbol{\delta}^{(\ell+1)}$:

$$\boldsymbol{\delta}^{(\ell)} \leftarrow \theta'\left(\mathbf{s}^{(\ell)}\right) \otimes \left[W^{(\ell+1)}\boldsymbol{\delta}^{(\ell+1)}\right]_1^{d^{(\ell)}}$$

---

**דוגמה.** נתונה רשת עם 3 שכבות $(L = 3)$ ונקודה $x = 2, \quad y = 1$




נבחין כי:

$$d^{(0)} = 1, \quad d^{(1)} = 2, \quad d^{(2)} = 1, \quad d^{(3)} = 1$$

נגדיר בשלב הראשון את $W^{(\ell)}$ לכל $\ell = 1, \ldots 3$:

$$W^{(1)} = \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{bmatrix}, \quad W^{(2)} = \begin{bmatrix} 0.2 \\ 1 \\ -3 \end{bmatrix}, \quad W^{(3)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

טבלת ה-forward propagation:

| $\mathbf{x}^{(0)}$ | $\mathbf{s}^{(1)} = \left(W^{(1)}\right)^T \mathbf{x}^{(0)}$ | $\mathbf{x}^{(1)}$ | $\mathbf{s}^{(2)} = \left(W^{(2)}\right)^T \mathbf{x}^{(1)}$ | $\mathbf{x}^{(2)}$ | $\mathbf{s}^{(3)}$ | $\mathbf{x}^{(3)}$ |
|---|---|---|---|---|---|---|
| $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ | $\begin{bmatrix} 0.1 & 0.3 \\ 0.2 & 0.4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0.7 \\ 1 \end{bmatrix}$ | $\begin{bmatrix} 1 \\ \tanh(0.7) \\ \tanh(1) \end{bmatrix} = \begin{bmatrix} 1 \\ 0.6 \\ 0.76 \end{bmatrix}$ | $\begin{bmatrix} 0.2 \\ 1 \\ -3 \end{bmatrix}^T \cdot \begin{bmatrix} 1 \\ 0.6 \\ 0.76 \end{bmatrix} = [-1.48]$ | $\begin{bmatrix} 1 \\ -0.90 \end{bmatrix}$ | $[-0.8]$ | $-0.66$ |

**כעת** נעבור ל- backpropagation

נשתמש בהגדרת $\theta = \tanh$ ונחשב את $\delta^{(3)}$:

$$\delta^{(3)} = 2\left(x^{(3)} - 1\right)\left(1 - \left(x^{(3)}\right)^2\right) = -1.855$$

נשתמש בלולאה:

**for** $\ell = L-1$ to $1$ **do**:

Let $\theta'\left(\mathbf{s}^{(\ell)}\right) = \left[1 - \mathbf{x}^{(\ell)} \otimes \mathbf{x}^{(\ell)}\right]_1^{d^{(\ell)}}$

והטבלה:

| $\boldsymbol{\delta}^{(3)}$ | $\boldsymbol{\delta}^{(2)}$ | $\boldsymbol{\delta}^{(1)}$ |
|---|---|---|
| $\left[-1.855\right]$ | $\left[\left(1-0.9^2\right)\cdot 2 \cdot -1.855\right] = [-0.69]$ | $\begin{bmatrix}-0.44\\0.88\end{bmatrix}$ |

וכעת חישובי השגיאות:

$$\frac{\partial e}{\partial W^{(1)}} = \mathbf{x}^{(0)}\left(\boldsymbol{\delta}^{(1)}\right)^T = \begin{bmatrix}-0.44 & 0.88\\-0.88 & 1.75\end{bmatrix}$$

$$\frac{\partial e}{\partial W^{(2)}} = \begin{bmatrix}-0.69\\-0.42\\-0.53\end{bmatrix}$$

$$\frac{\partial e}{\partial W^{(3)}} = \begin{bmatrix}-1.85\\1.67\end{bmatrix}$$

## 1.2 מרכיבים הכל ביחד

**אלגוריתם 2** חישוב $E_{in}(\mathbf{w})$ והגרדיאנט $g = \nabla E_{in}(\mathbf{w})$

**Input:** $\mathbf{w} = \left\{W^{(1)}, \ldots, W^{(L)}\right\}$; $\mathcal{D} = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$.
**Output:** error $E_{in}(\mathbf{w})$ and gradient $g = \left\{G^{(1)}, \ldots, G^{(L)}\right\}$.

(1) Initialize: $E_{in} = 0$, and $G^{(\ell)} = 0 \cdot W^{(\ell)}$ for $\ell = 1, \ldots, L$
(2) **for** each data point $(\mathbf{x}_n, y_n)$ , $n = 1, \ldots, N$ , **do**
  (a) Compute $\mathbf{x}^{(\ell)}$ for $\ell = 0, \ldots, L$ [forward propagation]
  (b) Compute $\boldsymbol{\delta}^{(\ell)}$ for $\ell = L, \ldots, 1$ [backpropagation]
  (c) $E_{in} \leftarrow E_{in} + \frac{1}{N}\left(\mathbf{x}^{(L)} - y_n\right)^2$
  (d) **for** $\ell = 1, \ldots, L$ **do**
    (i) $G^{(\ell)}(\mathbf{x}_n) = \left[\mathbf{x}^{(\ell-1)}\left(\boldsymbol{\delta}^{(\ell)}\right)^T\right]$
    (ii) $G^{(\ell)} \leftarrow G^{(\ell)} + \frac{1}{N}G^{(\ell)}(\mathbf{x}_n)$

כל התהליך הנ"ל נועד בשביל עדכון של **איטרציה אחת בלבד** - שיניב לנו סט משקלים חדש לכל שכבה באיטרציה הבאה:

$$W^{(\ell)}_{(t+1)} \leftarrow W^{(\ell)}_{(t)} - \eta G^{(\ell)}_{(t)}$$

## 1.3 בעיות נפוצות ושאלות פתוחות

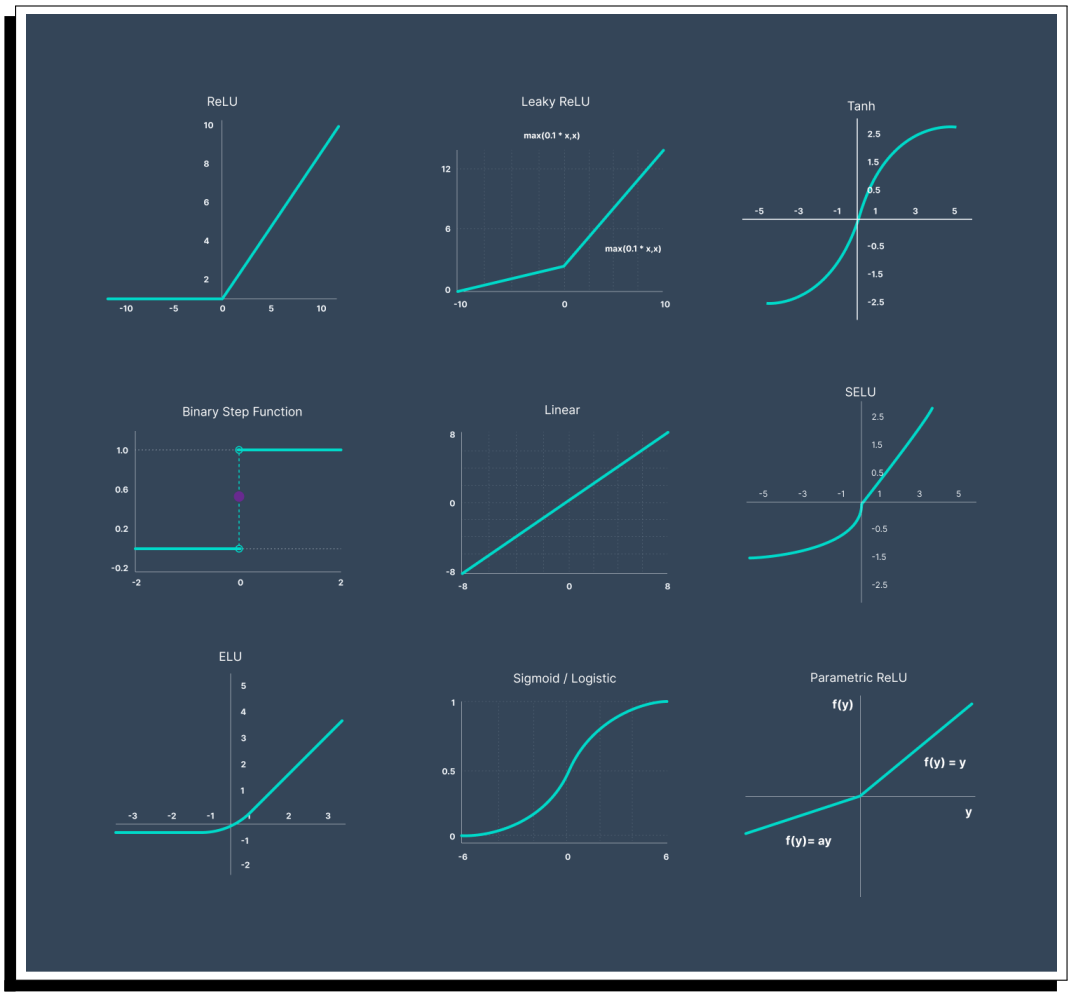
**בעיה.** מה עושים כאשר פונקצית המטרה שלנו לא קונבקסית? הרי יכול להיות ש GD יתכנס לנקודת מינימום מקומית

**פתרון.** במקום להתחיל מנקודה אחת את תהליך ה GD, נתחיל מכמה נקודות ולהשתמש בסט הולידציה על מנת להשוות בין החזאים השונים שקיבלנו.

הערה. אם נאתחל את המשקולות כאפסים $\left(W^{(\ell)}=0\right)$ אז לכל $\ell$ נקבל $\delta^{(\ell)}=0$ וכתוצאה הנגזרות של השגיאה יהיו 0 וסיים לאחר איטרציה אחת ולא נתקדם.

**בעיה.** **Vanishing Gradient** הוא מצב בו ערכי $|w_{ij}|$ יהיו מאוד גבוהים ובקצוות פונקצית האקטיבציה שלנו (נניח ב-tanh) נגיע לערכי $1$ ו- $-1$ וכתוצאה מכך הנגזרת של $\theta$ תתקרב מאוד ל-0 וכתוצאה מכך גם ערכי $\delta^{(\ell)}$.

**פתרון.** שימוש בפונקציות אקטיבציה שונות (כמו $ReLU$) או החלפת פונקצית האקטיבציה בשכבת המוצא בפונקציה לינארית




### 1.3.1 אז כיצד נאתחל את המשקלים?

מומלץ לאתחל את המשקלים בצורה **בלתי תלויה סטטיסטית** עם משתנה אקראי **נורמלי**, בעל **תוחלת** 0 **ושונות קטנה**. בצורה זו המשקלים יהיו סביב נקודת האפס, אך לא אפס ממש.

### 1.3.2 סיום האלגוריתם

נהוג לשלב מספר קריטריוני עצירה

- מספר איטרציות
- שינוי קטן יחסית בשגיאה
- השגיאה האבסולוטית קטנה מערך מסוים

# 2 Stochastic (Online, Sequential) Gradient Descent

תהליך ה- GD שדיברנו עליו עד כה נקרא **Batch Gradient Descent** והוא מחושב עבור **כל סט הדוגמאות** לפני עדכון המשקלים.

גרסה אחרת של GD היא גרסה סטוכאסטית שלו - **Stochastic Gradient Descent - SGD**

כמה נקודות על SGD:

- בכל פעם בוחרים באופן אקראי (אחיד) **דוגמה אחת** מסט הדוגמאות
- עבור הדוגמה הספציפית מחשבים את השגיאה ואת הגרדיאנט שמושפע מהדוגמה **הספיציפית הזו בלבד**
- בעזרת החישובים שלנו על הנקודה $\mathbf{x}_n$ הספציפית - נעדכן את סט הפרמטרים:

$$W^{(\ell)} = W^{(\ell)} - \eta G^{(\ell)}(\mathbf{x}_n)$$

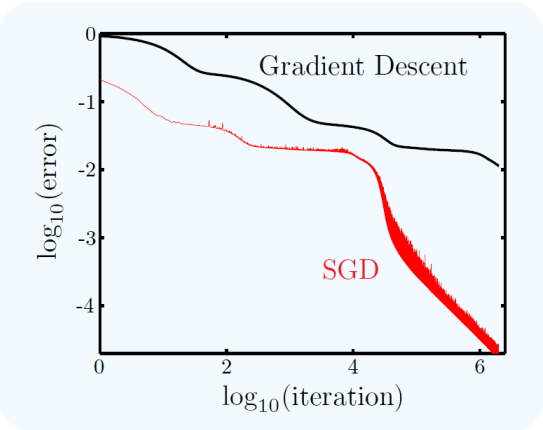
- נקבל עדכונים מאוד **רועשים** אשר לאורך האימון יבטלו אחד את השני ויתכנסו לתוצאה טובה כמו Batch Gradient Descent

$$\eta \cdot \frac{1}{N} \sum_{n=1}^{N} \nabla e_n(\mathbf{w})$$

- האלגוריתם יעיל יותר מאלגוריתם GD המקורי כיוון שאין צורך למצוא $N$ חישובי גרדיאנט, אלא רק אחד, משמע הוא יעיל פי $\frac{1}{N}$
- **הרעשים שמתקבלים מאופן הפעולה תורמים לנו בהתחמקות ממינימום מקומי!**
- אפשרי לשלב את 2 הטכניקות ולקחת תת-קבוצה של הדוגמאות ועליה לבצע GD

**דוגמה.** נתבונן בהבדלים בין Batch Gradient Descent ל- SGD

נתונות 500 דוגמאות, רשת בעלת 2 שכבות עם 5 שכבות חבויות, קצב למידה $\eta = 0.01$



# 3 Variable Learning Rate Gradient Descent

היריסטיקה פשוטה שעוזרת בתהליך ה GD היא שינוי ערכי קצב הלמידה $\eta$
בכל צעד נתבונן בשגיאה

- אם השגיאה קטנה - נגדיל את ערך $\eta$
- אם השגיאה לא קטנה - נחזור צעד אחורה ונקטין את ערך $\eta$

**אלגוריתם 3** Variable Learning Rate Gradient Descent

(1) Initialize $\mathbf{w}(0)$, and $\eta_0$ at $t = 0$. Set $\alpha > 1$ and $\beta < 1$.
(2) **while** stopping criterion has not been met **do**
  (a) Let $\mathbf{g}(t) = \nabla E_{in}(\mathbf{w}(t))$ , and set $\mathbf{v}(t) = -\mathbf{g}(t)$
  (b) **if** $E_{in}(\mathbf{w}(t) + \eta_t \mathbf{v}(t)) < E_{in}(\mathbf{w}(t))$ **then**
    (i) accept: $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta_t \mathbf{v}(t)$; $\eta_{t+1} = \alpha \eta_t$
  (c) **else**
    (i) reject: $\mathbf{w}(t+1) = \mathbf{w}(t)$; $\eta_{t+1} = \beta \eta_t$
  (d) Iterate to the next step, $t \leftarrow t + 1$

ונזכור:

$$\alpha \approx 1.05 - 1.1$$
$$\beta \approx 0.5 - 0.8$$

# 4 Steepest Descent

השאלה הנשאלת היא מדוע בכלל להסתבך עם קביעת $\eta$ אם אפשרי לעשות חישוב אנליטי? הרי אנחנו רוצים לרדת בכיוון המנוגד לגרדיאנט.

לכן, במקום לבחור $\eta$ כלשהו, נבחר $\eta$ אופטימלי שיביא למינימום את $E_{in}$

**אלגוריתם 4** Variable Learning Rate Gradient Descent

(1) Initialize $\mathbf{w}(0)$, and $\eta_0$ at $t = 0$.
(2) **while** stopping criterion has not been met **do**
  (a) Let $\mathbf{g}(t) = \nabla E_{in}(\mathbf{w}(t))$ , and set $\mathbf{v}(t) = -\mathbf{g}(t)$
  (b) Let $\eta^* = \text{argmin}_\eta E_{in}(\mathbf{w}(t) + \eta \mathbf{v}(t))$
  (c) $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta^* \mathbf{v}(t)$
  (d) Iterate to the next step, $t \leftarrow t + 1$

# 5 Regularization and Validation

- עד כה ראינו שרשתות נוירונים זהו כלי חזק מאוד שכולל שימוש ב GD ויכול לקרב אותנו לכל פונקציה שרק נרצה - משמע ישנה אופציה להתקל בהתאמת יתר (overfitting)
- גודל שכבת הכניסה אמנם אינו נתון לשינוי (גודל הקלט/פיצ'רים) וכך גם גודל שכבת היציאה (תלוי בבעיה אותה פותרים)
- מספר וגודל השכבות החבויות הוא פרמטר הניתן לשינוי
  - המחשבה הראשונית היא להקטין את מספר דרגות החופש שלנו ברשת (מספר השכבות החבויות והנוירונים)
  - מאידך מה שנהוג לעשות **זה בדיוק ההיפך** - לתת לרשת להיות כמה שיותר גדולה ולשלוט על התאמת היתר בעזרת רגולריזציה וולידציה

## 5.1 Weight Based Complexity Penalties

אחת שיטות הרגולריזציה המוכרות היא **Weight decay** - מה שמוכר לנו בתור Ridge Regression בבעית הרגרסיה ברשת הנוירונים נגדיר את השיטה בצורה דומה אך שונה

**הגדרה.** Weight squared decay - הוספת סכום ריבועי המשקולות בריבוע

$$E_{aug}(\mathbf{w}) = E_{in}(\mathbf{w}) + \frac{\lambda}{N}\sum_{\ell,i,j}\left(w_{ij}^{(\ell)}\right)^2$$

הרגולריזציה הנ"ל נקראת כך כיוון שבעת הגזירה נקבל איבר התלוי ב $W^{(\ell)}$:

$$\frac{\partial E_{aug}(\mathbf{w})}{\partial W^{(\ell)}} = \frac{\partial E_{in}(\mathbf{w})}{\partial W^{(\ell)}} + \frac{2\lambda}{N}W^{(\ell)}$$

ובעדכון המקדמים נכפיל בקצב הלמידה $\eta$ ובמינוס (כדי ללכת נגד כיוון הגרדיאנט) - מה שיגרור דעיכה (decay) לכיוון האפס.

## 5.2 Weight Elimination

תפקידה "להעלים" משקלים קטנים

**הגדרה.** Weight Elimination

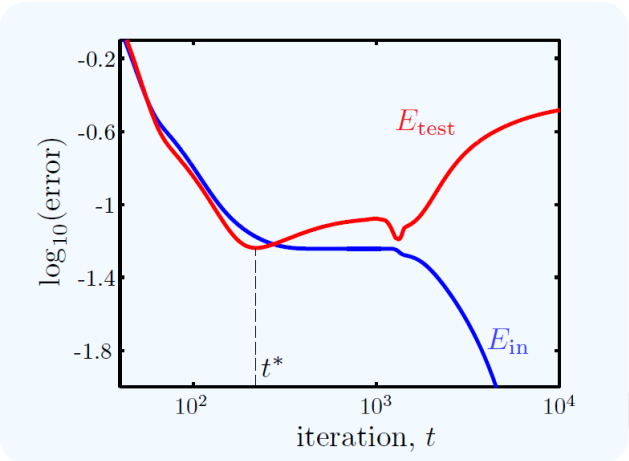
$$E_{aug}(\mathbf{w},\lambda) = E_{in}(\mathbf{w}) + \frac{\lambda}{N}\sum_{\ell,i,j}\frac{\left(w_{ij}^{(\ell)}\right)^2}{1+\left(w_{ij}^{(\ell)}\right)^2}$$

הנגזרת:

$$\frac{\partial E_{aug}}{\partial w_{ij}^{(\ell)}} = \frac{\partial E_{in}}{\partial w_{ij}^{(\ell)}} + \frac{2\lambda}{N}\cdot\frac{\left(w_{ij}^{(\ell)}\right)^2}{\left(1+\left(w_{ij}^{(\ell)}\right)^2\right)^2}$$

## Early Stopping 5.3

שיטת נוספת לרגולריזציה - כוללת הפעלת GD אך במקום לבצע המון איטרציות, עוצרים בנקודה בה השגיאה על סט הולידציה עולה.

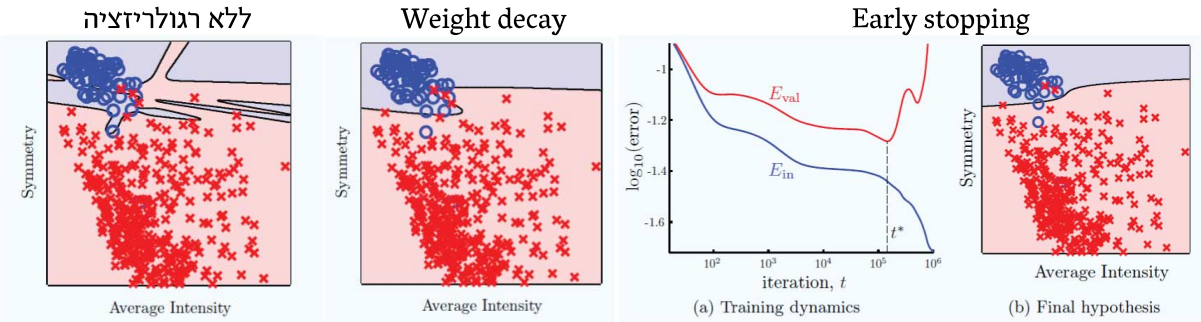



מהות שיטה זו לרגולריזציה מושתתת על העובדה **שמספר איטרציות מוגבל גורר מספר היפתוזות מוגבל** - משמע אנחנו לא נבדוק את כלל ההיפותזות שלנו, אלא נתמקד בתת-קבוצה של היפותזות!

הערה. שיטה זו קשורה מאוד לשיטת Weight decay כיוון שעצם העובדה שאנחנו מגבילים את האיטרציות גורמת לכך שהמשקולות תשארנה נמוכות וקרובות לאפס.

**דוגמה לשימוש ברגולריזציה**

500 דוגמאות, רשת נוירונים עם שכבה חבויה אחת בעלת 10 נוירונים, 2 פיצ'רים




| | $E_{train}$ | $E_{val}$ | $E_{in}$ | $E_{out}$ |
|---|---|---|---|---|
| No regularization | - | - | 0.2% | 3.1% |
| Weight decay | - | - | 1.0% | 2.1% |
| Early stopping | 1.1% | 2.0% | 1.2% | 2.0% |

# 6 Approximation vs Generalization

אפשר להראות שאם נבחר מספיק נוירונים עבור רשת בעלת 2 שכבות נוכל לקרב כל פונקציה שנרצה.
באופן כללי ניתן לרשום את החזאי של רשת כזו:

$$h(\mathbf{x}) = \theta\left(w_{01}^{(2)} + \sum_{j=1}^{m} w_{j1}^{(\ell)} \theta\left(\sum_{i=0}^{d} w_{ij}^{(1)} x_i\right)\right)$$

בצורה יותר נוחה:

$$h(\mathbf{x}) = \theta\left(w_0 + \sum_{j=1}^{m} w_j \theta\left(\mathbf{v}_j^T \mathbf{x}\right)\right)$$

מאידך, נוכל להשתמש בפרספטרון עם טרנס' לא לינארית ולבנות פרספטרון שתלוי לא לינארית בפיצ'רים שלנו ופונקצית $\theta$ לא לינארית ולקבל:

$$\mathbf{x} = \mathbf{\Phi}\left(\mathbf{x}\right) \to \mathbf{z} = [1, \phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_m(\mathbf{x})]^T$$
$$h(\mathbf{x}) = \theta\left(w_0 + \sum_{j=1}^{M} w_j \phi_j\left(\mathbf{x}\right)\right)$$

אך הדבר **השונה** הוא עצם העובדה שהטרנספורמציה מופעלת **אך ורק** על הפיצ'רים שלנו: $\phi_j\left(\mathbf{x}\right)$ לעומת רשת הנוירונים **שפועלת** גם על המשקולות $\theta\left(\mathbf{v}_j^T \mathbf{x}\right)$ - משמע בעזרת רשת הנוירונים אנחנו מתאימים את עצמנו **הרבה יותר טוב** למידע שלנו כי לומדים את ה-$w_{ij}$ בהתאם לדוגמאות לעומת הפרספטרון בו קבענו את $\phi_i$ עוד לפני האימון!