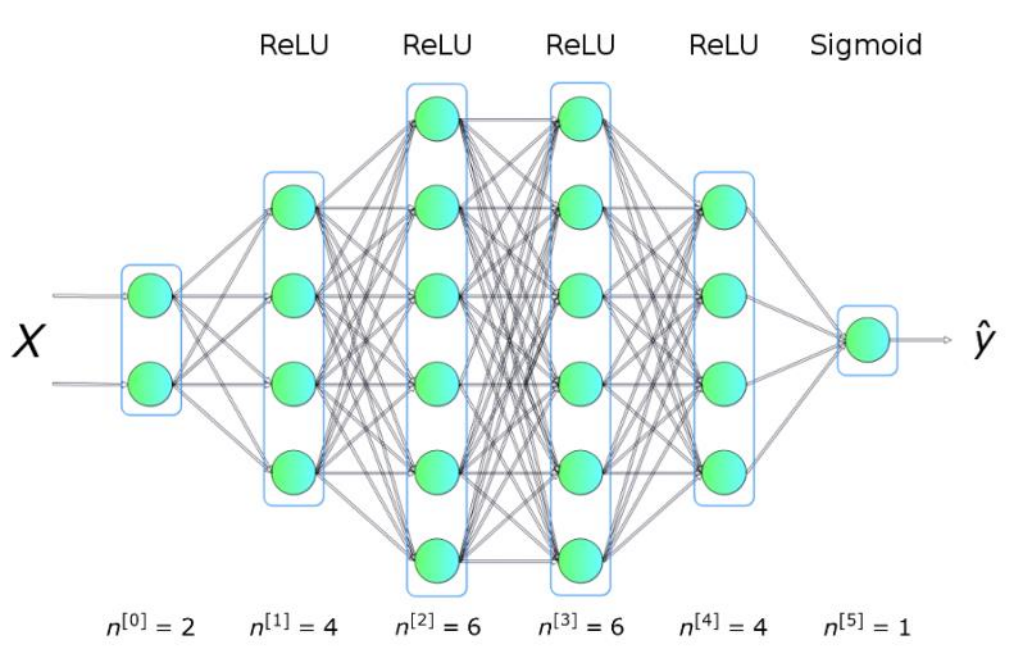
המתמטיקה שמאחורי - רשתות נוירונים/Neural Networks

11:20 AM Wednesday, September 4, 2024

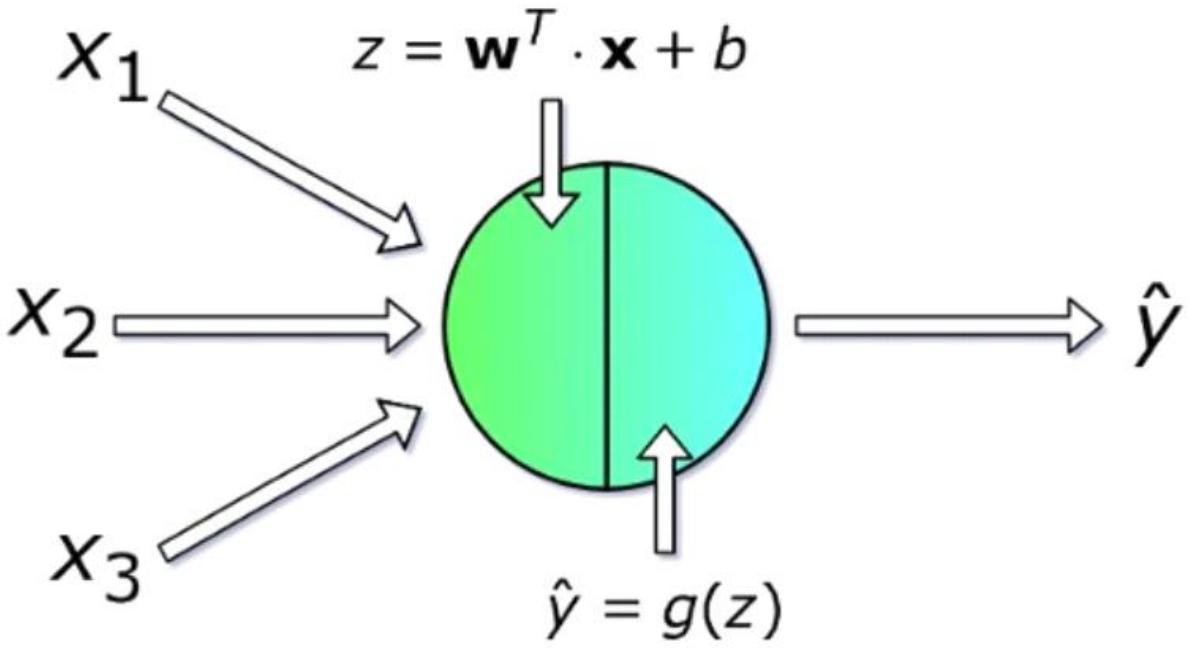
מה זה בכלל Neural Networks? -

זוהי שיטה בהשראה ביולוגית, (ההשוואה מתכוונת לנוירונים אשר נמצאים במוחנו) לבניית תוכנות מחשב המסוגלות ללמוד ולמצוא באופן עצמאי סוגי קשרים בנתונים. רשתות הן אוסף של 'נוירוני' תוכנה המסודרים בשכבות, המחוברים יחדיו באופן המאפשר תקשורת ביניהם.



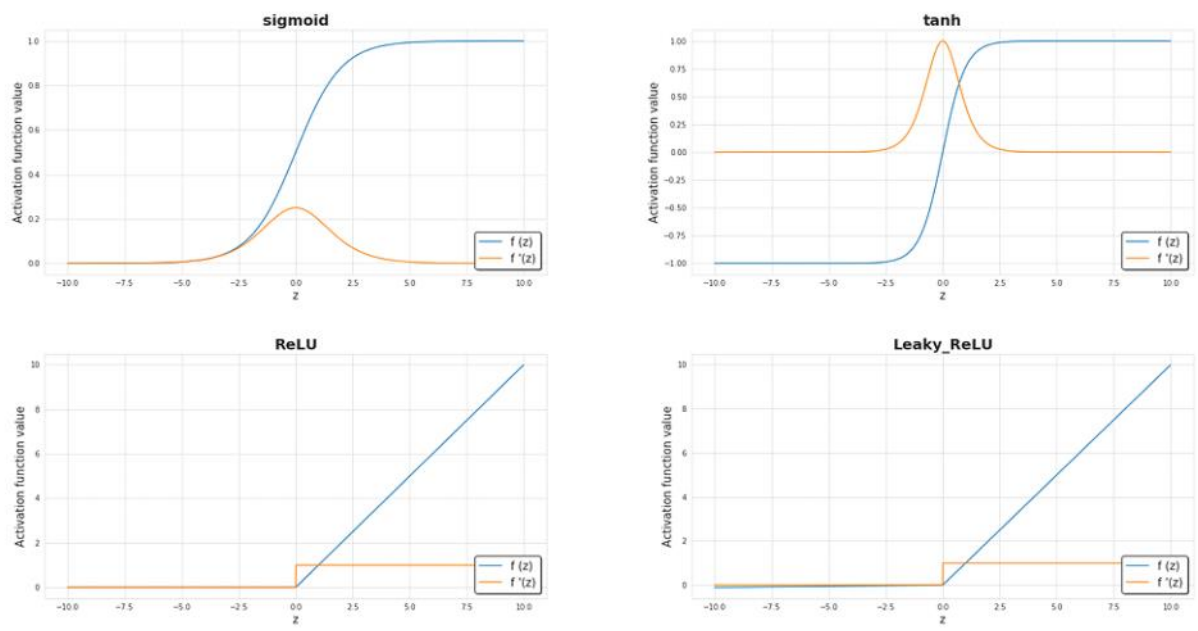
מה תפקידו של הנוירון הבודד? -

כל נוירון מקבל סט של ערכי x הממוספרים מ1 עד n כקלט, לפיהם הוא מחשב החיזוי של ערך ה-Hat-y.
כל x הוא וקטור אשר מכיל את ערכי ה-Feature באחת מ-m הדוגמאות מסט האימונים. לכל נוירון יש סט פרמטרים משל עצמו, אשר בדרך כלל נקראים w (עמודת הווקטור של המשקלים - Weights) ו-b (Bias). כאשר שתי אלו משתנים ומתכוננים כל הזמן במהלך תהליך הלמידה.
בכל איטרציה כל נוירון מחשב את ה-Weighted Average של ערכי הווקטור x, בהתבסס על ערכי הווקטור w ומוסיף לו Bias מסוים. התוצאה של החישוב עוברת דרך פונקציה לא לינארית g שעליה נסביר בהמשך.



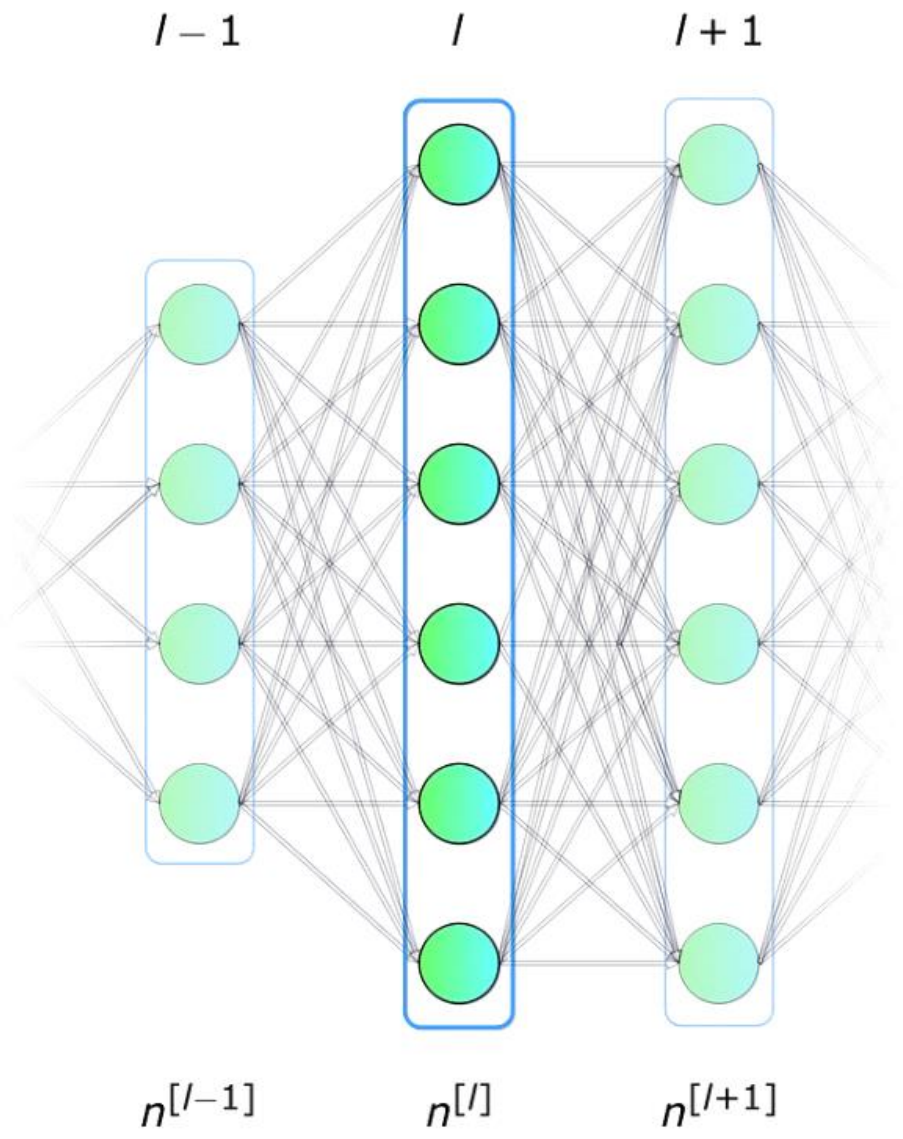
הפונקציה g ומה היא עושה? (Activation Function) -

Activation Functions הם אחת ממרכיבי המפתח של רשתות הנוירונים. בלעדיהם, רשת הנוירונים שלנו תהפוך לשילוב של פונקציות לינאריות, כך שבמהלך החישובים היא תהפוך לפונקציה לינארית בעצמה. במידה והיא הייתה כזאת הייתה לה יכולת התרחבות מוגבלת, שהיא לא יותר מרגרסיה לוגיסטית (התרחבות לפי קו השיפוע ולא יותר מזה). ה-Activation Functions מאשר גמישות רבה בתהליך הלמידה ומאפשרת לנו ליצור פונקציות מורכבות יותר במהלך תהליך הלמידה. מה גם שיש לה השפעה משמעותית לטובה על מהירות הלמידה של המודל שלנו. במידה ואנחנו עוסקים בסיווג בינארי ואנו רוצים שהערכים המוחזרים מהמודל יהיו בטווח שבין 0 ל-1.
התמונה הבאה מציגה כמה מה-Activation Functions הנפוצות. בניהם RELU, Sigmoid, Tanh וכו' ...



שכבת נוירונים בודדת -

עכשיו בוא נראה מה קורה עם חישובי הנוירונים ברמת כל שכבה של רשת נוירונים. אנחנו נשתמש בידע של החישובים הקורים בכל נוירון ונבצע וקטור על פני שכבה מלאה כדי לשלב את החישובים הללו לתוך משוואות של מטריצות. כדי לאחד את הסימון, המשוואות ייכתבו עבור השכבה הנבחרת [l]. כאשר i, מסמן את האינדקס/המיקום של הנוירון בשכבה המדוברת.



הערה חשובה נוספת: כשכתבנו את המשוואות לנוירון בודד, השתמשנו ב-x וב-y-Hat, שהיו בהתאמה וקטור העמודה של ה-Features והערך החזוי. כאשר עוברים לסימון הכללי עבור שכבה, אנו משתמשים בווקטור a - כלומר הפעלה של השכבה המתאימה. לכן וקטור x הוא ההפעלה עבור שכבה 0 - שכבת הקלט.

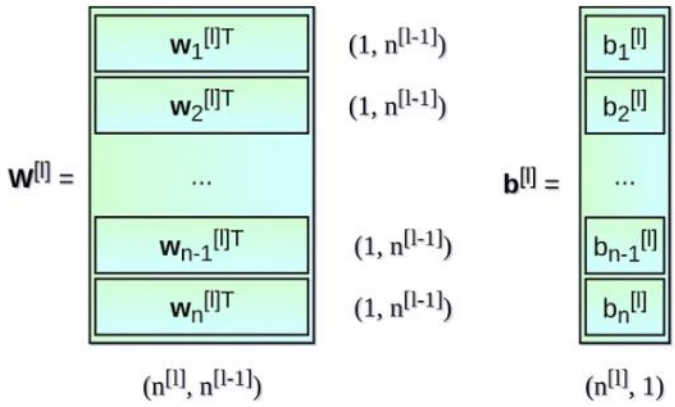
כל נוירון בשכבה מבצע חישוב לפי המשוואות הבאות:

$$z_i^{[l]} = \mathbf{w}_i^T \cdot \mathbf{a}^{[\mathbf{l-1}]} + b_i \qquad a_i^{[l]} = g^{[l]}(z_i^{[l]})$$

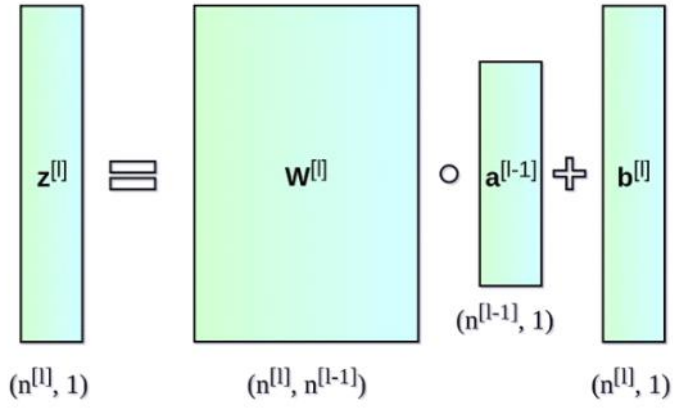
לדוגמא, החישוב של השכבה ה-2 (כאשר מחובר אליה בשכבה שלפניה נוירון בודד) היא:

$$
\begin{aligned}
z_1^{[2]} &= \mathbf{w}_1^T \cdot \mathbf{a}^{[1]} + b_1 & a_1^{[2]} &= g^{[2]}(z_1^{[2]}) \\
z_2^{[2]} &= \mathbf{w}_2^T \cdot \mathbf{a}^{[1]} + b_2 & a_2^{[2]} &= g^{[2]}(z_2^{[2]}) \\
z_3^{[2]} &= \mathbf{w}_3^T \cdot \mathbf{a}^{[1]} + b_3 & a_3^{[2]} &= g^{[2]}(z_3^{[2]}) \\
z_4^{[2]} &= \mathbf{w}_4^T \cdot \mathbf{a}^{[1]} + b_4 & a_4^{[2]} &= g^{[2]}(z_4^{[2]}) \\
z_5^{[2]} &= \mathbf{w}_5^T \cdot \mathbf{a}^{[1]} + b_5 & a_5^{[2]} &= g^{[2]}(z_5^{[2]}) \\
z_6^{[2]} &= \mathbf{w}_6^T \cdot \mathbf{a}^{[1]} + b_6 & a_6^{[2]} &= g^{[2]}(z_6^{[2]})
\end{aligned}
$$

כפי שניתן לראות, עבור כל אחת מהשכבות עלינו לבצע מספר פעולות דומות מאוד. השימוש ב-For-Loop למטרה זו אינו יעיל במיוחד, ולכן כדי לזרז את החישוב של המשוואות נשתמש בוקטוריזציה. קודם כל, על ידי הערמה של וקטורים אופקיים של משקלים w (Transported) נבנה מטריצה W. באופן דומה, נערום יחד את כל ה-Bias של כל נוירון לכל ווקטור אנכי הנקרא b. כעת ניתן לבנות משוואות הבנויות ממטריצות יחידות אשר מאפשרות לנו לבצע חישוב של כל הנוירונים הנמצאים בשכבה בפעם אחת.

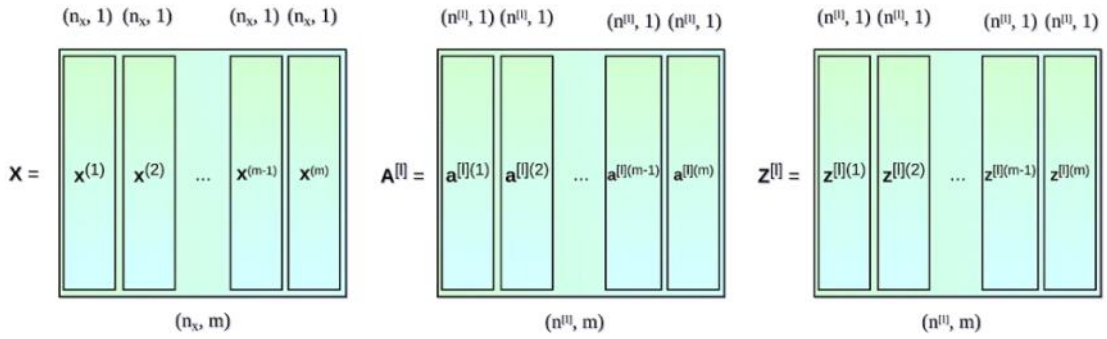


$$
\mathbf{z}^{[l]} = \mathbf{W}^{[l]} \cdot \mathbf{a}^{[l-1]} + \mathbf{b}^{[l]} \qquad \mathbf{a}^{[l]} = g^{[l]}(\mathbf{z}^{[l]})
$$



וקטוריזציה על פני מספר דוגמאות -

המשוואה שעשינו עד כה כוללת רק נוירון אחד בודד a^1 בשכבה הקומת לו. במהלך תהליך הלמידה של רשתות נוירונים, בדרך כלל עובדים עם סטים עצומים של נתונים, עד מיליוני ערכים. לכן נשתמש בוקטוריזציה על פני מספר דוגמאות. נניח שבמערכת הנתונים שלנו יש M ערכים עם Features n_x כל אחת. ראשית, מרכיבים את הוקטורים האנכיים x, a ו-z של כל שכבה ויוצרים את המטריצות X, A ו-Z, בהתאמה. לאחר מכן נכתוב מחדש את המשוואה שהוקמה קודם לכן, תוך התחשבות במטריצות החדשות שנוצרו.



$$\mathbf{Z}^{[l]} = \mathbf{W}^{[l]} \cdot \mathbf{A}^{[l-1]} + \mathbf{b}^{[l]} \qquad \mathbf{A}^{[l]} = g^{[l]}(\mathbf{Z}^{[l]})$$

פונקציית הפסד (Loss Function) -

מקור המידע הבסיסי המציג את ההתקדמות של תהליך הלמידה הוא הערך של פונקציית ההפסד (ה-Loss Function). פונקציית ההפסד נועדה להראות עד כמה אנחנו רחוקים מהפתרון ה'אידיאלי' (לדוגמא במקרה של חיזוי מספר לפי ציור, הפתרון האידאלי הוא המספר אליו אנו מתכוונים כאשר אנחנו מציירים אותו). במקרה שלנו השתמשנו ב-Binary Crossentropy, אך בהתאם לבעיה ניתן להשתמש בפונקציות אחרות. הפונקציה המשמשת אותנו מתוארת בנוסחה הבאה, והשינוי בערכה במהלך תהליך הלמידה מוצג בסרטון, הוא מראה כיצד בכל איטרציה הערך של פונקציית ההפסד יורד והדיוק עולה.

$$J(W,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$
$$L(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y}))$$

Accuracy
... And Loss

כיצד רשתות נוירונים לומדות? -

תהליך הלמידה עוסק בשינוי ערכי הפרמטרים W ו-b כך שפונקציית ההפסד תהיה ממוזערת (תשאף ל-0). על מנת להשיג את המטרה הזאת, נחשב את נקודת המינימום של הפונקציה באמצעות Gradient Descent על מנת למצוא את המינימום של הפונקציה.
Gradient זוהי שיטה למציאת נקודות מינימום ומקסימום מקומיים תוך כדי ביצוע סדרה של "צעדים" ההולכים ומתכנסים לנקודות שאנו מחפשים (במקרה שלנו נקודת המינימום של הפונקציה), וזאת באמצעות הנגזרת של הפונקציה והשיפוע המוביל למורד הפונקציה.
בכל איטרציה נחשב את הערכים של הנגזרות החלקיות של ה-Loss Function שלנו ביחס לכל אחד מהפרמטרים של רשת הנוירונים שלנו. (לנגזרת יש יכולת לתאר את השיפוע של הפונקציה שלנו)
באמצעות ה-Gradient Descent אנו יודעים לנוע במורד הגרף על מנת להגיע אל נקודת המינימום.
בסרטון הבא ניתן לראות כיצד באמצעות ה-Gradient Descent אנו מתקרבים לנקודת המינימום עד כאשר אנו מגיעים אליה (כמו התגלשות במגלשה):

Gradient
... Decent In

ברשתות נוירונים, מוצאים את נקודת המינימום באותה הדרך כמו בסרטון. אך ההבדל היחידי ברשתות נוירונים היא שיש לנו הרבה יותר פרמטרים לתמרן בניהם. אז איך ניתן לחשב נגזרות כאלו מורכבות?

התפשטות לאחור (Backpropagation) -

Backpropagation הוא אלגוריתם המאפשר לנו לחשב שיפוע מסובך מאוד כמו של רשתות נוירונים. הפרמטרים של רשת הנוירונים מותאמות לפי הנוסחאות הבאות.

$$\mathbf{W}^{[l]} = \mathbf{W}^{[l]} - \alpha \mathbf{dW}^{[l]}$$
$$\mathbf{b}^{[l]} = \mathbf{b}^{[l]} - \alpha \mathbf{db}^{[l]}$$

במשוואות למעלה, α מייצג את קצב הלמידה - הוא בעצם Hyperparameter המאפשר לנו לשלוט בערך של הצעדים שניקח כל איטרציה לכיוון נקודת המינימום. בחירת קצב הלמידה היא קריטית, כאשר אנחנו מגדירים את α בערך נמוך מדי, המודל שלנו ילמד לאט מאוד, כאשר אנחנו מגדירים את α בערך גבוה מדי נוכל לעקוף כל פעם את המינימום ולא להגיע אליה כראוי.
dW ו-db מחושבים באמצעות כלל השרשרת, נגזרות חלקיות של ה-Loss Function ביחס ל-W ו-b. הגודל של dW ו-db זהה לזה של W ו-b בהתאמה. התמונה הבאה מציגה את רצף הפעולות שמופעלות בתוך רשת הנוירונים. אנו רואים בבירור כיצד התפשטות קדימה ואחורה פועלות יחד כדי לייעל את ה-Loss Function.

$$\mathbf{dW}^{[l]} = \frac{\partial L}{\partial \mathbf{W}^{[l]}} = \frac{1}{m}\mathbf{dZ}^{[l]}\mathbf{A}^{[l-1]T}$$
$$\mathbf{db}^{[l]} = \frac{\partial L}{\partial \mathbf{b}^{[l]}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{dZ}^{[l](i)}$$
$$\mathbf{dA}^{[l-1]} = \frac{\partial L}{\partial \mathbf{A}^{[l-1]}} = \mathbf{W}^{[l]T}\mathbf{dZ}^{[l]}$$
$$\mathbf{dZ}^{[l]} = \mathbf{dA}^{[l]} * g'(\mathbf{Z}^{[l]})$$