Model Building with:

PyTorch

הטכניון
היחידה ללימודי המשך
ולימודי חוץ ע"ש עזריאלי

TECHNION
Azrieli Continuing Education and
External Studies Division

- פונקציות אקטיבציה
- עקרונות מנחים לבנית מודל

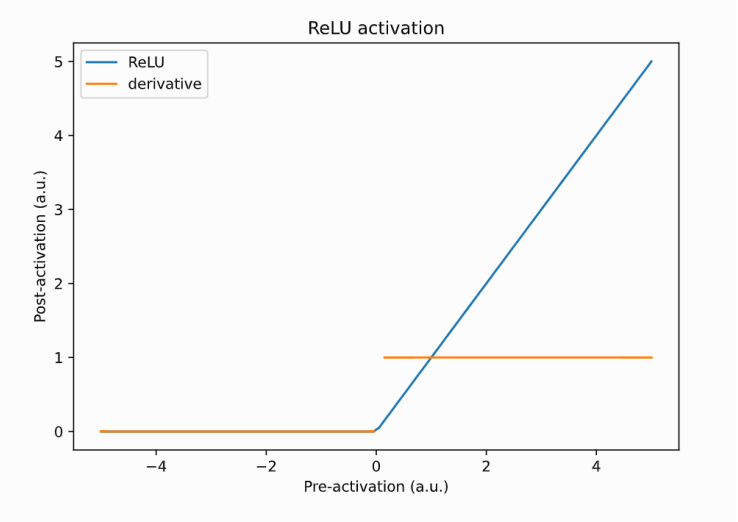
ReLU (Rectified Linear Unit):

- **מתי להשתמש:**
  - בעיות רגרסיה או סיווג עם **נתונים לא לינאריים**, כאשר הערכים השליליים לא קריטיים.
  - רשתות עמוקות – מונע בעיית התעממות שיפוע – Gradient vanishing
- **חיסרון:** נוירונים עלולים "למות" (להישאר עם ערך אפס) אם המשקל מתעדכן לערך שלילי קבוע.

•מתאים לפונקציית עלות:

- **Cross Entropy** בבעיות סיווג או **MSE**, בבעיות רגרסיה.

Rectified Linear Unit (Relu)



$$f(x) = max\ (0, x)$$

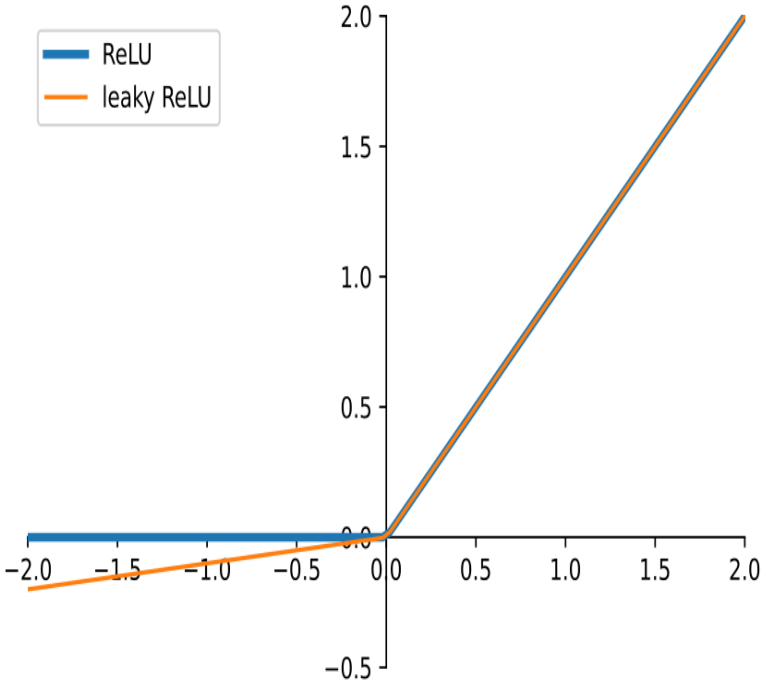
Leaky ReLU:

עקרון פעולה:

בניגוד לReLU שמחזיר 0 עבור כל ערך שללי, פה יוחזרו 10% מהערך השלילי כך שתהיו בו התחשבות , אבל קטנה.

- **מתי להשתמש:**
  - כשיש חשש מנוירונים "מתים" ב.ReLU-
  - מתאים למקרים בהם חשוב לשמור קצת מידע בערכים שליליים.
- **חיסרון :** יכול לגרום לשינויים קטנים ועדינים יותר בעדכון משקלים.
- **מתאים לפונקציית עלות:**
  - **Cross Entropy** או **MSE** בדומה לReLU

Leaky Rectified Linear Unit



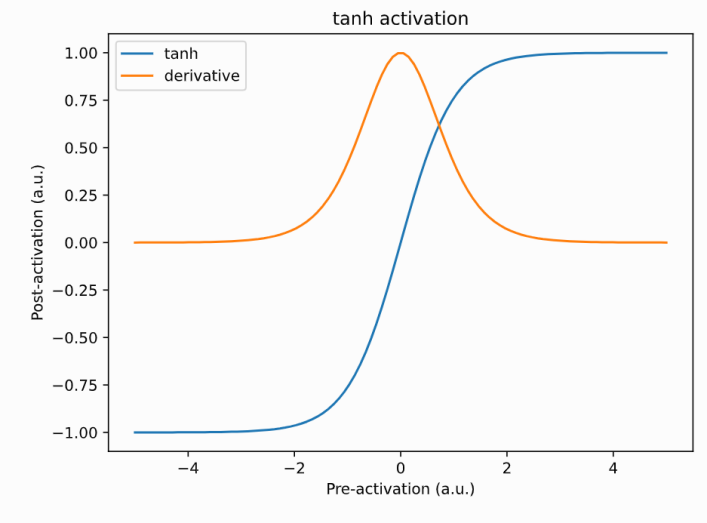
Leaky ReLU

$$f(x) = max\ (0.1x, x)$$

Tanh (Hyperbolic Tangent):

- **מתי להשתמש:**
  - בעיות עם נתונים סימטריים
  - כאשר ערכים שליליים חשובים ללמידה.
  - מתאים יותר לבעיות רגרסיה גליות\מחזוריות.
- **חיסרון:**
  - גורם להתעממות גרדיאנט בערכים קיצוניים.
  - דורש למידת משקלים מדויקת כדי לא לגרום להתכנסות איטית.
- **מתאים לפונקציית עלות:**
  - MSE או Cross Entropy, כאשר הערכים נעים בין -1 ל-1.

Hyperbolic Tangent

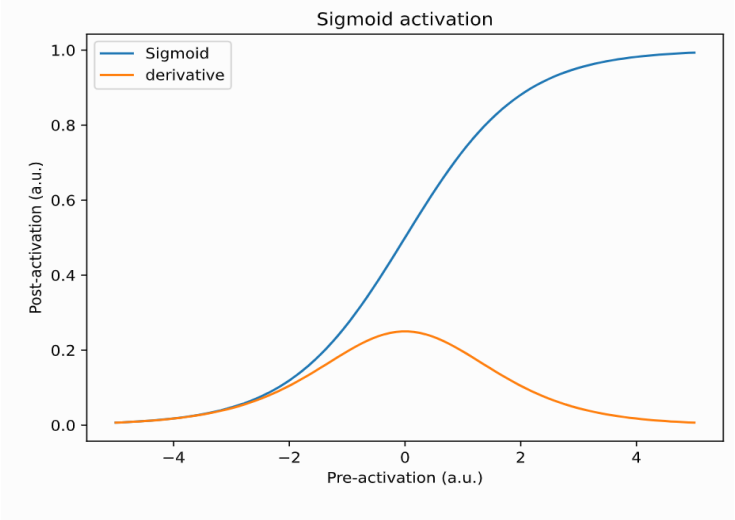


$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

Sigmoid:

- **מתי להשתמש:**
  - סיווג בינארי, כאשר את רוצה הסתברות בין 0 ל-1.
  - בעיות בהן נדרשת נורמליזציה לפלט סופי.
- **חיסרון:**
  - עלול לגרום להתעממות גרדיאנט.
  - לא מתאים לרשתות עמוקות עם שכבות נסתרות רבות.
- **מתאים לפונקציית עלות:**
  - **Binary Cross Entropy (BCE)** לסיווג בינארי.
  - **BCEWithLogitsLoss**כאשר רוצים למנוע בעיות מספריות.

Sigmoid Function



$$S(x) = \frac{1}{1 + e^{-x}}$$

פונקציית אקטיבציה	יתרונות	חסרונות	מתאים לסוג הבעיה	טווח פלט
ReLU	מהיר, מונע Gradient Vanishing	נוירונים "מתים"	בעיות כלליות לא לינאריות	$(0,\infty)$
Leaky ReLU	מונע נוירונים מתים	עדכון איטי בערכים שליליים	כמו + ReLU שמירת מידע שלילי	$(-\infty,\infty)$
Tanh	שומר ערכים חיוביים ושליליים	Gradient Vanishing אפשרי	נתונים סימטריים	$[-1,1]$
Sigmoid	הסתברות בין 0 ל-1	Gradient Vanishing, איטי	סיווג בינארי	$[0,1]$

- **עקרון הפשטות –להתחיל ממודל קטן**
  - לדוגמה עם שכבה נסתרת אחת או שתיים בלבד.
  - לשים מספר קטן של נוירונים, לדוגמה: 16, 32 או 64.
  - אם המודל לא מצליח ללמוד, אפשר להוסיף שכבות או נוירונים בהדרגה
  - מספר הנוירונים יכול לקטון בהדרגה (32 – 16 – 8)
  - לרוב 2-4 שכבות מספיקות
  -

- **מספר פיצ'רים כמדריך**
  - כמות הנוירונים בשכבה הראשונה יכולה להיות קרובה למספר הפיצ'רים בנתונים.
  - לדוגמה, אם יש 10 פיצ'רים, אז שכבה ראשונה עם 16 או 32 נוירונים תוכל להתאים.

- **הערכת הלימוד**
  - **Underfitting:** המודל לא לומד טוב (טעות גבוהה גם באימון וגם בבדיקה).

    כדאי לנסות להוסיף מורכבות (עוד שכבות נסתרות או עוד נוירונים.)
  - **Overfitting:** המודל לומד טוב מדי על הנתונים באימון, אבל טועה הרבה בבדיקה.

    כדאי להקטין את המורכבות
- **התאמת שכבת האקטיבציה לסוג הבעיה**
  - בעיות רגרסיה לא לינאריות דורשות יותר שכבות כדי לקלוט את הדפוס הלא לינארי.
  - לדוגמה, עבור נתוני **גל סינוסי** ,שכבה אחת עם פונקציית **Tanh** יכולה לעבוד מצוין .