Andrew and Erna Viterbi Faculty
of Electrical Engineering
Electronics
Computers
Communications

Technion
Israel Institute of Technology

SIPL

הטכניון – מכון טכנולוגי לישראל

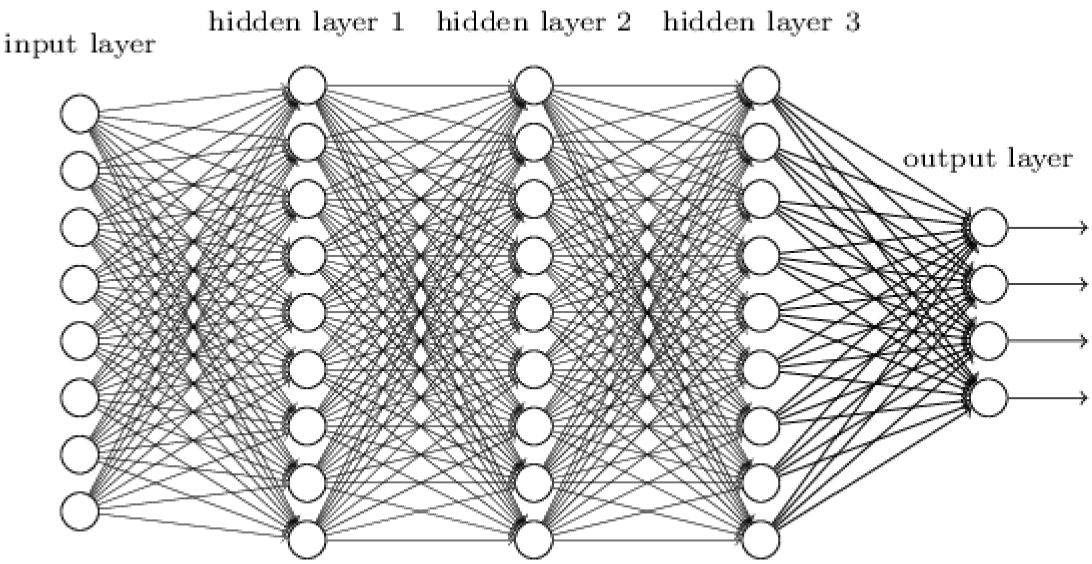
הפקולטה להנדסת חשמל ע״ש ויטרבי

המעבדה לעיבוד אותות ותמונות

מעבדה בהנדסת חשמל 2/3/4

מבוא ללמידה עמוקה

Introduction to Deep Learning



נכתב ע״י: נדב בהונקר ושונית חביב, ספטמבר 2018.

עדכונים: אורי בריט – יולי 2019, אפריל 2020.

אתר המעבדה: http://sipl.technion.ac.il

גרסה נוכחית: APR2020.2

תוכן עניינים

רקע למעבדה – מפגש ראשון

מטרת הניסוי

היכרות עם אלגוריתמי למידה חישובית ממשפחת רשתות הנוירונים. אלגוריתמים אלה הוכיחו את עצמם בשנים האחרונות כבעלי ביצועים גבוהים וכעת הם משמשים ככלי יישומי בהרבה תחומים ותעשיות. בניסוי נבנה את היסודות כדי להבין את כיצד האלגוריתם פועל ונממש רשת נוירונים. מכיוון שרשתות נוירונים הינן אלגוריתמים בתוך התחום הנקרא מערכות לומדות, יש צורך בידע של מושגים בסיסיים בתחום.

רקע על מערכות לומדות[1]

מערכות לומדות (Machine Learning) הוא תחום העוסק בפיתוח ותכנון אלגוריתמים המאפשרים מיצוי אוטומטי של מידע מתוך נתונים אמפיריים. לפי אחת ההגדרות, מערכת לומדת היא מערכת אשר משפרת את ביצועיה בביצוע משימה נתונה ככל שהיא מבצעת משימה זו.

לתחום יישומים רבים ומגוונים: זיהוי כתב יד ודיבור, סיווג מסמכים, לימוד במשחקים, תרגום, רובוטיקה, זיהוי פנים ועצמים בתמונה, חיזוי פיננסי ועוד.

בניגוד לפתרון אלגוריתמי "מסורתי", בו האלגוריתם מפורש וקבוע וכל פרטי הפתרון ידועים למתכנן, אלגוריתם לומד מוכתב עד כדי מאפיינים תלויי מידע, המתכוונים במהלך הלימוד.

לגישה לומדת יתרונות רבים, ביניהם הקניית יכולות שהן מעבר ליכולת הניתוח של מפתח המערכת, והסתגלות לסביבה משתנה. כמובן שיחד עם היתרונות הרבים קיימים חסרונות, ביניהם הצורך בכמות גדולה של מידע מתויג באופן ספציפי ואיכותי וכוח החישוב היקר הנדרש.

בעיית הסיווג

נציג כעת את בעיית הסיווג תוך התייחסות למקרה מסוים:

1. מקרה פשוט

במפעל אריזת דגים מעוניינים להפריד באופן אוטומטי בין הדגה היומית. בפרט, יש להפריד בין דגי הסלמון (salmon) לדגי הלברק (sea bass) על סמך תמונה של הדג, דהיינו למצוא מסווג שמוציא עבור כל תמונה פלט המציין את סוג הדג.

למשימה מסוג זה קודם שלב של מיצוי מאפיינים "מעניינים" מתוך התמונה. מאפיינים אלו יסייעו לנו בסיווג הדגים. הוחלט כי ימוצו מתוך התמונה שני המאפיינים הבאים: אורך הדג ובהירות הדג (בהינתן שהתמונה נתונה ברמות אפור). נציין שמיצוי המאפיינים מתוך התמונה דורש הפעלת תהליכי עיבוד תמונה על-מנת לנקות את התמונה מרעש ולבצע סגמנטציה של הדג מתוך הרקע. במעבדה זו לא נעסוק בשלב זה ונניח כי בידינו שני המאפיינים הרצויים עבור כל תמונה.

[1] החומר מתבסס על הרקע לניסוי "מבוא למערכות לומדות" של המעבדה לבקרה, רובוטיקה ולמידה חישובית.

2. <u>הגדרות</u>

בבעיית הסיווג אנו נדרשים לתכנן מסווג באמצעות סט דוגמאות מתויגות כך שיסווג בצורה הטובה ביותר קלט חדש.

נשתמש בהגדרות הבאות לתיאור בעיית הלמידה :

- מרחב הקלט : $X \in \mathbb{R}^D$, כך שכל דוגמה $x = (x^1, x^2, \dots, x^D) \in X$. כאשר $D > 1$, נקרא ל- X "וקטור המאפיינים".

  במקרה שלנו : $X \in \mathbb{R}^2$, כך שעבור כל דוגמה $x \in X$ יש שני מאפיינים : אורך ובהירות.

- מרחב הפלט : $\Omega = \{1, 2, \dots, C\}$ מכיל את אוסף המחלקות האפשריות.

$\Omega = \{-1, +1\}$ מכיל במקרה זה שתי מחלקות לסיווג : סלמון ולברק. בעיית סיווג מסוג זה, עם שתי מחלקות בלבד, נקראת בעיית סיווג בינארית.

- מסווג : העתקה $f : X \to \Omega$ אשר נותנת תיוג לכל קלט במרחב הקלט.

  הפונקציה אותה נרצה ללמוד היא זו המתייגת כל תמונה אם מדובר בדג סלמון או דג הלברק.

- סדרת הלימוד (training set) : סט של n דוגמאות מתויגות (labeled) $\{x_i, y_i\}_{i=1}^{n}$, כאשר $y_i \in \Omega$ הוא הסיווג הנכון של תבנית הקלט.

  במקרה שלנו : תמונות דגים אשר תויגו למחלקות הנכונות באופן ידני.

- סדרת הבוחן (test set) : סט של m דוגמאות (**שאינו חלק מסדרת הלימוד**) $\{x_i, y_i\}_{i=n+1}^{n+m}$. סט זה משמש להערכת הביצועים של המסווג הנלמד. בשלב הערכת הביצועים נפעיל את המסווג על וקטורי המאפיינים $\{x_i\}_{i=n+1}^{n+m}$ ונשווה את התיוגים של המסווג לתיוגים הנכונים $\{y_i\}_{i=n+1}^{n+m}$.

  במקרה שלנו : תמונות נוספות של דגים שתויגו ידנית בהם לא השתמשנו לאימון המסווג.

באופן כללי, נהוג לחלק את בעיות הלמידה לשניים : בעיות סיווג ובעיות רגרסיה. ההבדל היחידי בין השניים הוא מרחב הפלט. בבעיות סיווג הוא בדיד, ובבעיות רגרסיה הוא רציף. לדוגמה, בבעיית סיווג הדגים המחלקות הן הדגים : סלמון או לברק. לו היינו מנסים לקבוע את *משקל הדג* מתוך התמונה, הרי זו הייתה בעיית רגרסיה, שכן הפלט הוא מספר רציף.

אלגוריתמים רבים ללמידה בבעיות סיווג ניתן להמיר בצורה פשוטה לבעיות רגרסיה, וכן כל העקרונות שילמדו במעבדה זו לגבי רשתות נוירונים תקפים גם לגבי בעיות רגרסיה.

לרוב נניח כי סדרת הלימוד וסדרת הבוחן שלנו הן בעלות אותו פילוג ובלתי תלויות (i.i.d. = Independent and identically distributed random variables). כלומר :

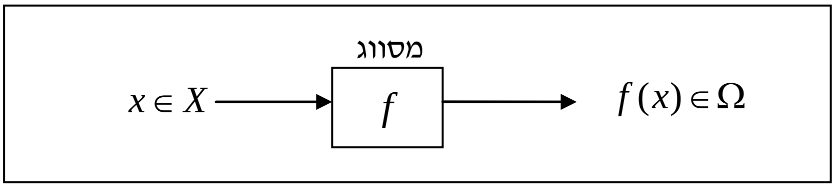
$$p(x_i) = p(x_j), \qquad \forall x_i, x_j \in \{training \backslash test\ set\}$$

$$p(x_i, x_j) = p(x_i)p(x_j), \qquad \forall x_i, x_j \in \{training \backslash test\ set\}$$

נגדיר שנית את בעיית הסיווג תוך שימוש במושגים הנ״ל:

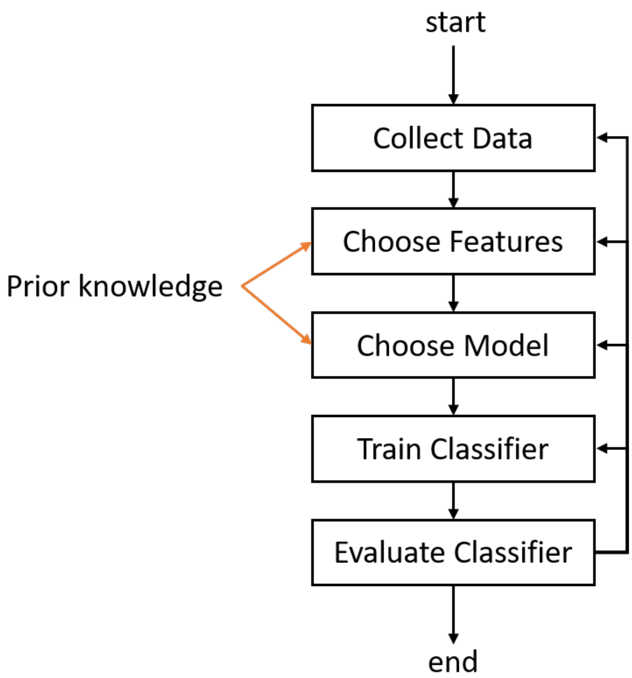
בהינתן סדרת לימוד $\{x_i, y_i\}_{i=1}^{n}$, נרצה למצוא מסווג $f: X \to \Omega$ כך שיסווג את סדרת הבוחן $\{x_k\}_{k=n+1}^{n+m}$ למחלקה המתאימה עם שגיאה קטנה ככל הניתן.

באופן סכמתי מסווג מוגדר בצורה הבאה:



הלמידה המתוארת הינה למידה אינדוקטיבית: הכללה מהפרט – סדרת הלימוד, אל הכלל – קלט חדש. בפרט נשים לב כי נדרש לתכנן מסווג בעל שגיאה קטנה על דוגמאות חדשות שלא שייכות לסט הדוגמאות בו נעזרנו לתכנון.

התרשים הבא מתאר בצורה סכמתית את תהליך הלימוד בבעיית הסיווג.



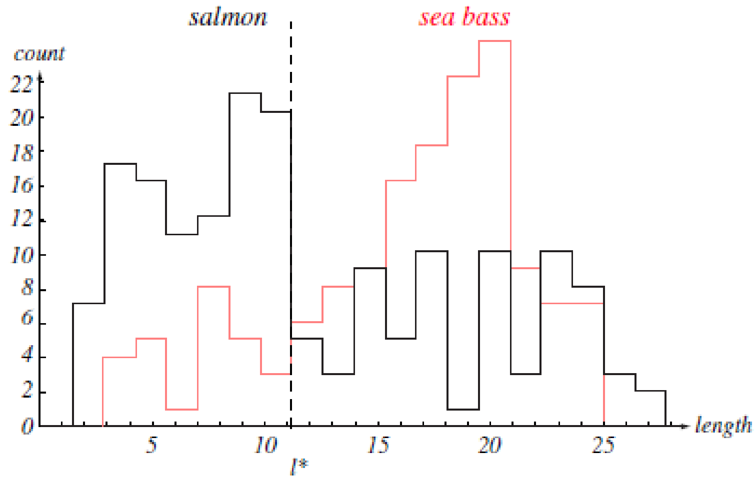
איור 1 – תיאור סכמתי של תהליך הלימוד

לאחר שלב איסוף המידע (תמונות של דגים עם תיוג), יש לבחור את המאפיינים הרלוונטיים לתיאורו ואת המודל המתאים למערכת (אורך ובהירות); בשלב זה ניתן לשלב ידע מקדים אודות המערכת. לאחר מכן מגיע שלב אימון המסווג ובחינת ביצועיו.

הידע המקדים אותו אנחנו מכניסים למערכת בדוגמת הדגים הינה שניתן להסתפק בבהירות ואורך הדג על מנת לקבוע את סוגו.

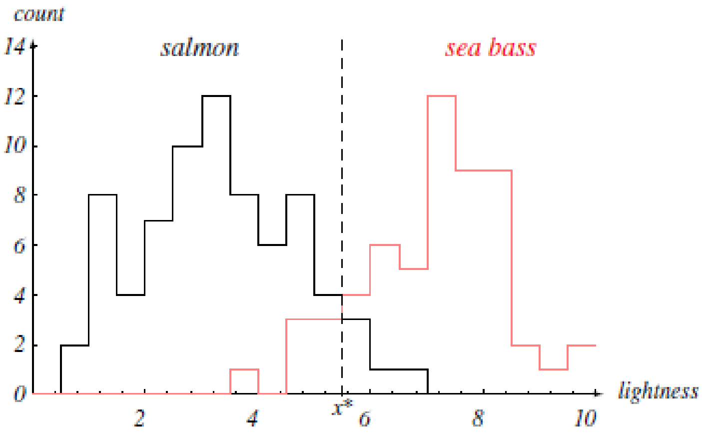
בשלב ראשון נתבונן בדוגמאות שלנו, ובהתפלגות המאפיינים השונים.

באיור 2 מוצגת ההיסטוגרמה של הדוגמאות ע״פ המאפיין של אורך הדג[2]. ניתן לראות כי אין הפרדה טובה בין המחלקות מכיוון שהחפיפה גדולה מידי.



איור 2 – היסטוגרמה ע״פ אורך הדג

באיור 3 ניתן לראות את ההיסטוגרמה של הדוגמאות ע״פ המאפיין של בהירות הדג. כאן תחום החפיפה קטן יותר שכן ברוב המקרים הלברק בהיר יותר.



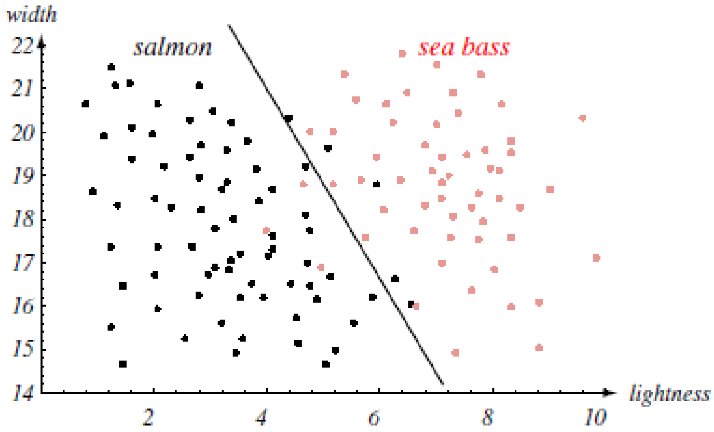
איור 3 - היסטוגרמה ע״פ בהירות הדג

[2] כל האיורים נלקחו מתוך:

Pattern Classification (2nd ed.), Richard O. Duda, Peter E. Hart and David G. Stork (John Wiley and Sons, 2001)

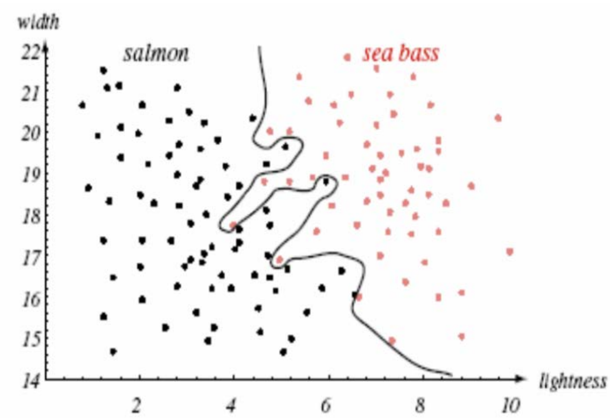
ונערכו לצורך התאמתם לתוכן החוברת.

נשאלת השאלה האם שימוש בשני המאפיינים יחד יכול להביא לייצוג טוב יותר של הבעיה. על מנת לענות על שאלה זו, ניתן לצייר את ערך המאפיינים בתרשים דו-ממדי, כפי שמוצג באיור 4. כפי שניתן לראות, ניתן להעביר קו ישר בין שתי המחלקות המסווג באופן נכון את רוב דוגמאות סדרת הלימוד. ניתן להבחין שבעזרת הייצוג הדו-ממדי שתי המחלקות ניתנות להפרדה טובה יותר מאשר בשימוש באחד מהמאפיינים.

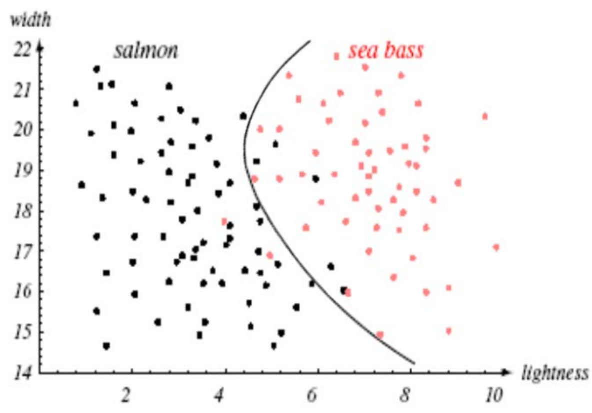


איור 4 – הצגה דו-ממדית של המאפיינים עם מסווג לינארי

כמובן שקיימות אפשרויות נוספות להעברת הקו לסיווג המחלקות. שתי דוגמאות מוצגות בהמשך. באיור 6 ניתן להבחין כי מושג סיווג מושלם על סדרת האימון אך אזורי ההחלטה מאוד מסובכים. אזורי החלטה כאלו יכולים להשיג אפס שגיאה על סדרת האימון, אך עלולים להוביל לשגיאה גדולה יותר על דוגמאות חדשות. באיור 5 מוצג מסווג בעל אזור החלטה מסובך יותר מאשר זה שבאיור 4, אך עם פחות שגיאות על סדרת האימון. בתכנון מסווג יש להתייחס ל-tradeoff בין סיבוכיות המודל והביצועים על סדרת האימון.



איור 6 – מסווג הגורם להתאמת יתר



איור 5 – מסווג פולינומיאלי

התופעה המתרחשת באיור 5 היא תופעה נפוצה ומוכרת כאשר משתמשים במסווגים מורכבים בעלי פרמטרים רבים. תופעה זו נקראת overfitting, וישנן מספר אינדיקציות לקיומה בהינתן מסווג מסוים.

3. <u>מדד ביצועים</u>

נשתמש במדד ביצועים כדי למדוד את מידת ההצלחה של המסווג שלנו f . נעריך את שגיאת המסווג באמצעות סדרת הבוחן (test set) $\{x_i, y_i\}_{i=n+1}^{n+m}$:

$$Err(f) = \frac{1}{m}\sum_{i=n+1}^{n+m} l(f(x_i), y_i)$$

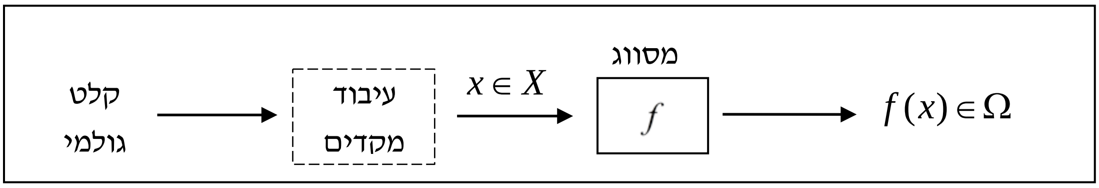
כאשר ℓ הינה פונקציית השגיאה (נקראת zero-one loss) : $\ell\left(f(x_i), y_i\right) = \begin{cases} 0 & f(x_i) = y_i \\ 1 & f(x_i) \neq y_i \end{cases}$.

שגיאת המסווג הינה השגיאה האמפירית על סדרת הבוחן. נרצה ללמוד מסווג f כזה שהשגיאה $Err(f)$ תהיה קטנה ככל האפשר.

4. <u>תהליך התכן של המסווג</u>

אלגוריתם הסיווג אינו מופעל בדרך כלל על הקלט הגולמי. קלט זה יעבור שלבים של עיבוד מקדים לפני למידת המסווג והפעלתו.

באופן סכמתי, נוסיף את השלב של העיבוד המקדים לתכן המסווג :



במקרה של הדגים לעיל, חלק מהעיבוד המקדים היה שלב מיצוי המאפיינים של אורך ובהירות הדג מתוך התמונות. שלב העיבוד המקדים נעשה בהתאם לאופי הקלט הגולמי ולסוג אלגוריתם הלמידה (אופי המסווג).

עיבוד מקדים של קבוצת הלימוד חיוני לטובת :

1. התאמת הקלט למודל הלמידה. כלומר, ישנם סוגי מסווגים המתאימים לקלט מסוג מסוים.
2. הורדת ממדיות הבעיה והפיכתה לפשוטה יותר.
3. האצת שלב הלמידה או האימון של המסווג.
4. שיפור רמת הביצועים של המסווג.

Gradient Descent

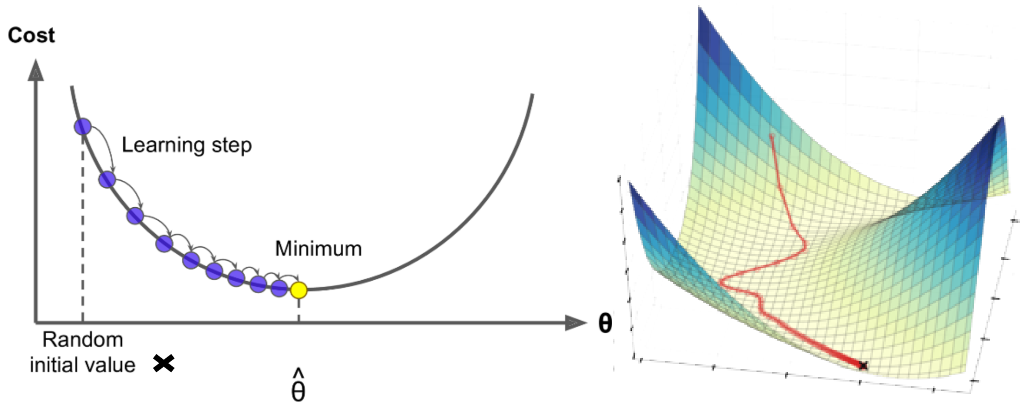
Gradient descent הינה שיטה איטרטיבית כללית למציאת מינימום של פונקציה. הרעיון מאוד פשוט: בהינתן פונקציה גזירה $f(x)$, אותה נרצה למזער, נעדכן את x לפי הנוסחה:

$$x_{k+1} = x_k - \eta_k g_k$$

$$g_k = f'(x_k)$$

כלומר, נזיז את הערך של x לכיוון בו הגרדיאנט הוא מינימלי (נגד כיוון הגרדיאנט). כאשר η הינו מקדם הנקרא *גודל הצעד* או *קצב הלמידה*. גודל הצעד יכול להישאר קבוע או להשתנות בתהליך הלמידה.

ויזואלית, התהליך עשוי להיראות כך:



איור 7 – המחשה ויזואלית של תהליך אימון באמצעות Gradient Descent

ניתן להיעזר באנלוגיה הבאה לקבלת אינטואיציה:

דמיינו את המצב ההיפותטי בו מטייל נמצא על הר בו שורר ערפל כבד כך שהוא לא מצליח לראות את תוואי השטח. המטייל מנסה לרדת מההר (אל הנקודה הנמוכה). מכיוון שאין ראות, הדרך בה יגיע למטה היא ע"י בחינת הכיוון בו הירידה תלולה ביותר והליכה בכיוון זה מספר צעדים קבוע בצורה איטרטיבית.

באנלוגיה:

- המיקום של המטייל מסומן ע"י x.
- תוואי השטח היא הפונקציה $f(x)$ שרוצים למזער.
- גודל הצעד של המטייל הוא גודל הצעד η.
- כיוון הירידה הוא גודל הגרדיאנט.

הערות:

- כדי שנוכל להפעיל את האלגוריתם, כל שנדרש הוא שהפונקציה $f(x)$ תהיה גזירה. אין מגבלה על מימד הפונקציה או על צורתה.
- האלגוריתם הפשוט ביותר למציאת מקסימום\מינימום של פונקציה שומר על גודל צעד קבוע. ישנן שיטות מתקדמות יותר: כאלו שמשנות את גודל הצעד בצורה אדפטיבית, כאלו שמשתמשות בקירוב מסדר שני של הפונקציה (ההסיאן) ועוד.

- אין הבטחה שהאלגוריתם יתכנס למינימום הגלובלי! ייתכן מאוד שנתכנס למינימום מקומי. הבטחת התכנסות למינימום גלובלי מתקיימת רק אם $f(x)$ פונקציה קמורה.
- תחום המחקר של מציאת מקסימום\מינימום של פונקציות נקרא *אופטימיזציה* וקיימים מספר קורסים בטכניון בהם מלמדים את הנושא לעומק.

רשתות נוירונים

כעת נציג את המודל הכללי של רשתות נוירונים. המודל מתאר את (כמעט) כל רשתות הנוירונים שקיימות. מהרשתות הפשוטות ביותר שנראה בהמשך, ועד רשתות גדולות ומורכבות שמשמשות חברות מסחריות (זיהוי פרצופים, תרגום וכו').

למען ההקדמה, נסתכל על רשת נוירונים כעל קופסה שחורה. נרצה שקופסה זו תהיה מסוגלת ללמוד מדוגמאות קיימות, ולאחר מכן להשתמש בידע שרכשה על מנת לבצע הסקה והחלטה על דוגמה חדשה. נתאר את הכניסות והיציאות למערכת שלנו, ואת הפרמטרים הנלמדים:

כניסות:

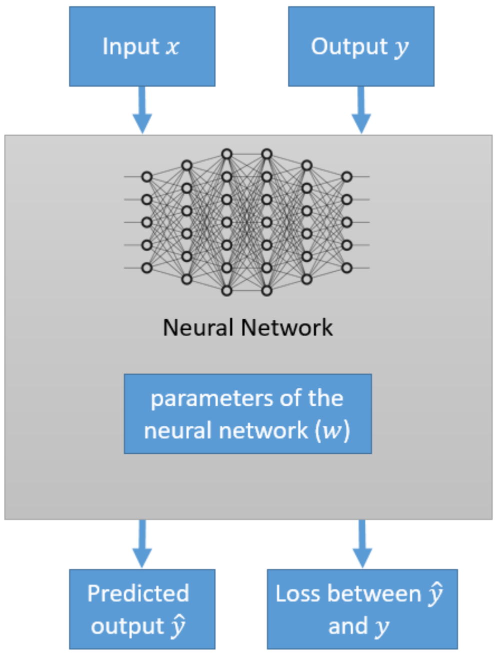
- x נקודת מידע כלשהי (תמונה, נתונים וכו')
- y תיוג מתאים לנקודה x

יציאות:

- $\hat{y}$ חיזוי של תיוג לנקודה x
- l ה-Loss של החיזוי $\hat{y}$ ביחס לתיוג האמיתי y

פרמטרים נלמדים:

- פרמטרי רשת הנוירונים המשמשים לחיזוי התיוג. נסמנם ב-w, ונקרא ל-w "וקטור המשקלים".



איור 8 - המחשה ויזואלית של רשת נוירונים כקופסה שחורה בעלת כניסות ויציאות

בשלב האימון של רשת הנוירונים, המערכת תקבל באופן סדרתי זוגות $\{x, y\}$, ותעדכן את הפרמטרים שלה w, על מנת להתאים את המוצא החזוי $\hat{y}$ להיות מתאים ככל הניתן ל-y. כלומר, נרצה לעדכן את פרמטרי המערכת במטרה **למזער ככל הניתן את שגיאת החיזוי**.
לאחר שהמערכת אומנה, נרצה לבצע הסקה על דוגמה חדשה. נזין לכניסת המערכת את הדוגמה הרלוונטית x, והמערכת תשתמש בפרמטרים אשר נלמדו בשלב הקודם על מנת לחזות $\hat{y}$ עבור x.
באופן זה ניתן לבצע חיזוי עבור דוגמאות שאנו לא יודעים את תיוגן, בעזרת למידה מדוגמאות מתויגות נתונות.

נוירונים?
האלגוריתמים המכונים רשתות נוירונים במקור נוצרו *בהשראת* המוח. באיזה מובן? במוח ישנם נוירונים (תאי עצב), יחידות עיבוד קטנות בעלות מספר קלטים ומוצא בודד אשר ממשקלות באופן שונה את הכניסות באופן נלמד (סינפסות).
התהליך שמתרחש ברשתות נוירונים אמיתיות במוח שונה ורחוק מהותית ממה שמתרחש כיום ברשתות נוירונים מלאכותיות ויש להתייחס לאנלוגיה בזהירות.

נסתכל על רשתות נוירונים כעל פונקציה $f: X \to \Omega$. באמצעות Gradient Descent נמצא את המינימום של פונקציית השגיאה בין חיזוי הרשת לבין התיוג האמיתי של כל דוגמה.
בהתייחס לאיור 1 – תיאור סכמתי של תהליך הלימוד, שלב בחירת המודל נקבע ע"י בחירת **פונקציית השגיאה** ובחירת **מבנה (ארכיטקטורת)** הרשת. שלב אימון המודל נעשה ע"י Gradient Descent כדלקמן:

<u>אלגוריתם האימון של רשת נוירונים</u>

<u>קלט</u>: סדרת דוגמאות מתויגות לאימון $\{x_i, y_i\}_{i=1}^{n}$, פרמטר קצב הלימוד $\eta_0 > 0$, מספר חזרות מקסימאלי על סדרת הלימוד k.

<u>פלט</u>: פונקציה f המבוטאת באמצעות אוסף הפרמטרים w.

<u>שלבי הלימוד</u>:

1. אתחול: אתחל את הווקטור w^0 לערך כלשהו.
2. עבור על דוגמאות סדרת האימון $i = 1,2,3 \dots , n$:
   - 2.1. קבל את הווקטור x_i והתיוג y_i.
   - 2.2. קבע את קצב הלימוד η_t.
   - 2.3. חשב את שגיאת המודל עם וקטור המשקלים הנוכחי w^t: $f(x, y, w^t)$.
   - 2.4. חשב את וקטור הגרדיאנט $g(x, y, w) = \nabla_w f(x, y, w)$.
   - 2.5. עדכן את וקטור המשקולות w^{t+1}:

$$\mathrm{w}^{t+1} = w^t - \eta_t g(x, y, w)$$

3. חזור על שלב 2 עד שמתקיים אחד משני תנאי העצירה הבאים:
   - 3.1. הושגה שגיאה נמוכה מספיק על סט המבחן.
   - 3.2. הושלמו k חזרות (epochs) על סדרת הלימוד.
4. החזר את אוסף הפרמטרים w.

האלגוריתם לעיל מתאר את פעולת רשת הנוירונים באופן כללי. במהלך המעבדה נראה מספר מקרים פרטיים מפורסמים שיתנו לכם טעימה מעולם הרשתות.

סיווג באמצעות שגיאה ריבועית ונוירון בודד - ADALINE

כאמור, כל הרשתות הן מקרים פרטיים של התהליך שמתואר לעיל. כעת נציג את המודל הפשוט ביותר.
בספרות האלגוריתם ידוע בשמות (Adaptive Linear Neuron) ADALINE, Widrow-Hoff או אלגוריתם LMS (Least Mean Squares).

פונקציית השגיאה שנרצה למזער היא שגיאת הסיווג האמיתית או המקורבת ע"י חישוב השגיאה על סט האימון:

$$Err(f) = \frac{1}{n}\sum_{i=1}^{n} l(f(x_i), y_i)$$

כאשר ℓ הינה פונקציית השגיאה (נקראת zero-one loss): $\ell\left(f(x_i), y_i\right) = \begin{cases} 0 & f(x_i) = y_i \\ 1 & f(x_i) \neq y_i \end{cases}$.

מה הבעיה בשימוש בפונקציית שגיאה זו?

שגיאה מסוג זה אינה גזירה! לכן לא נוכל להשתמש ב-Gradient Descent.

הפתרון: שימוש בפונקציית שגיאה חליפית (Surrogate Loss Function). כלומר נמזער פונקציה גזירה אחרת, שמזעורה יוביל למזעור שגיאת ה- zero-one.

פונקציית השגיאה שנבחר היא שגיאה ריבועית:

$$Err(f) = \frac{1}{n}\sum_{i=1}^{n} (f(x_i) - y_i)^2$$

כאשר אנחנו מניחים ש- $y_i \in \{-1,1\}$.

ארכיטקטורת הרשת תהיה פונקציה אפינית (שלעיתים מכונה, באופן מבלבל, כשכבה לינארית[3]):

$$f(x_i) = w^T x_i + b = \sum_{d=1}^{D} w_d x_i^d + b$$

$w = (w_1, w_2, \dots, w_D)$ הינו וקטור המשקולות של הפונקציה הלינארית ו- b הינו פרמטר ההטיה (bias).

הערה על סימונים: $x_i =$ הדוגמה ה-i בסט האימון, x_i^d – המאפיין ה-d של הדוגמה ה-i בסט האימון.

צורת רישום חלופית: ניתן להוסיף איבר נוסף לקלט $x^{D+1} = 1$ ולהגדיר $w_{D+1} = b$, המאפשרים שימוש בכתיב מקוצר:

$$f(x_i) = w^T x_i = \sum_{d=1}^{D+1} w_d x_i^d$$

[3] פעולה לינארית משמעותה כפל בווקטור (מטריצה). פעולה אפינית משמעותה כפל בווקטור (מטריצה) והוספת סקלר (וקטור).

כלומר, ע"י הגדלת גודל הקלט ב-1 והגדרת מספר קבוע בו (המספר אחד), והגדלת וקטור המשקולות ב-1 והגדרת איבר זה כ-b, הצלחנו להפוך את הביטוי המקורי w^Tx+b לביטוי הכולל אך ורק מכפלה יחידה w^Tx. שימו לב ששינוי הרישום לא משנה את העובדה שהפונקציה אפינית.

לאיזו משפחה שייך המשתנה $f(x_i)$? $\mathbb{R}$? $\mathbb{N}$?

$f(x_i)$ הוא סכום ממושקל של משתנים רציפים, לכן גם הוא רציף. איך בכל זאת נקבע את תיוג המחלקה מתוך מוצא רציף?

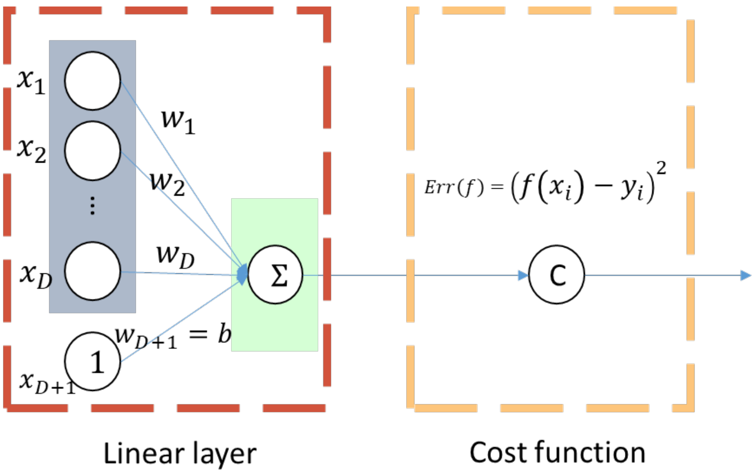
כאשר נבצע הסקה באמצעות המודל (כלומר נשאל על דוגמה חדשה מהי המחלקה שלה) נתייג אותה באמצעות פונקציית Hard Limiter (הידועה גם בשמות פונקציית מדרגה ו-heaviside), שמותאמת *לסיווג בינארי* בין שתי מחלקות:

$$\hat{y}=\varphi_{HL}(w^Tx)=\begin{cases}-1, & w^Tx<0\\ +1, & w^Tx\geq 0\end{cases}$$

נתבונן בפונקציה $g(x)=w^Tx$. המשוואה $g(x)=0$ מגדירה משטח החלטה המפריד בין שתי מחלקות: $g(x)<0$ ו-$g(x)\geq 0$, ובהתאם לכך נקבע הסיווג שנסמנו $\hat{y}$.

כלומר משטח ההחלטה הינו *על-מישור* (על-מישור=מישור ביותר מ-2 מימדים). כאשר $x\in\mathbb{R}^2$ אנחנו מקבלים קו לינארי.

ניתן לתאר סכמטית את המודל באופן הבא:



איור 9 – סכמה של מודל ADALINE

1. <u>אימון באמצעות Gradient Descent</u>

כאמור, נעדכן באופן איטרטיבי את וקטור המשקולות במטרה להקטין את שגיאת הסיווג על סדרת הדוגמאות. וקטור המשקולות באיטרציה t יסומן w^t.

על מנת לעדכן את הרשת יש לחשב את הגרדיאנט של המודל:

$$\frac{\partial Err(f)}{\partial w}=\frac{\partial(f(x)-y)^2}{\partial w}=\frac{\partial(w^Tx-y)^2}{\partial w}=2x(w^Tx-y)\propto x(w^Tx-y)$$

אלגוריתם אימון ADALINE

קלט: סדרת דוגמאות מתויגות לאימון $\{x_i, y_i\}_{i=1}^{n}$, פרמטר קצב הלימוד $\eta_0 > 0$, מספר חזרות מרבי על סדרת הלימוד K, פרמטר שינוי קצב הלימוד m.

פלט: וקטור פרמטרים $w^{t_{end}} = w^k = (w_1, w_2, \dots, w_{D+1})$.

שלבי הלימוד:

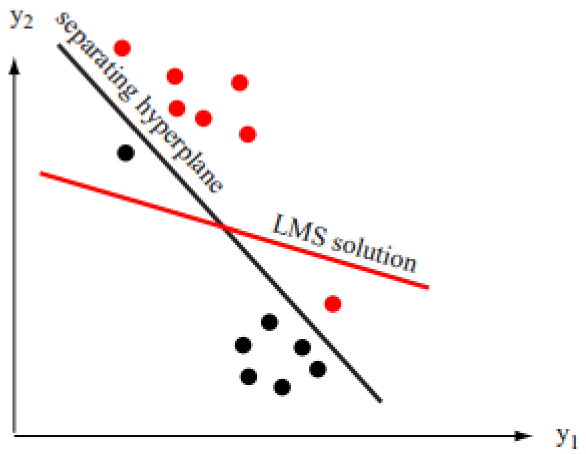
1. אתחול: אתחל את הווקטור w^0 בערך כלשהו.
2. עבור על דוגמאות סדרת האימון $i = 1,2,3, \dots, n$:
   - 2.1. קבל את הווקטור $x \in \mathbb{R}^{D+1}$ והתיוג $y \in \{-1,+1\}$ של הדוגמה ה-i.
   - 2.2. חשב את יציאת הרשת עם וקטור המשקלים הנוכחי w^0: $\hat{y} = {w^t}^T x$.
   - 2.3. קבע את קצב הלימוד: (עבור מימוש בסיסי $m = 1$)

     $$\eta_t = \eta_0 / m$$
   - 2.4. עדכן את וקטור המשקלים w^{t+1} (שהוא וקטור המשקלים אשר התקבל לאחר איטרציה t):

     $$w^{t+1} = w^t - \eta_t x(\hat{y} - y) = w^t - \eta_t x\left({w^t}^T x - y\right)$$
3. חזור על שלב 2 עד שהושלמו K חזרות על סדרת הלימוד (epochs).
4. החזר את w^k.

הערות:

- האלגוריתם לעיל הוא מקרה פרטי של האלגוריתם שראינו קודם לאימון רשתות.
- במקרה הפשוט הנ"ל היה ניתן לפתור את מערכת המשוואות ע"י נוסחה סגורה במקום התהליך האיטרטיבי. החיסרון של שימוש בנוסחה סגורה היא שלא ניתן להשתמש בה כאשר מספר הדוגמאות שלנו גדול מאוד.
- האלגוריתם לאו דווקא יתכנס לפתרון שנותן סיווג מושלם. לדוגמה:



איור 10 – דוגמה לפתרון של ADALINE (LMS) שלא נותן סיווג מושלם, לעומת פתרון של מודל אחר שכן נותן סיווג מושלם

סימולציה - בעיית סיווג האירוסים

דוגמה מפורסמת של בעיה שאותה ניתן לפתור באמצעות אלגוריתם לומד, הינה בעיית סיווג מיני אירוסים עפ״י אורך ורוחב עלי הכותרת ועלי הגביע.

על מנת לפשט את הבעיה, נממש את אלגוריתם ADALINE על סמך שני מאפיינים (אורך ורוחב עלי הגביע) ושני מיני אירוסים (Veriscolor ו-Setosa).

פתחו ב-Matlab את הקבצים adaline.m ו-iris_adaline והשלימו בהם את שורות הקוד.

*מומלץ לממש חלק זה לפני המשך הקריאה של חומר ההכנה.

תזכורת – התפלגות ברנולי

זוהי התפלגות של הטלת מטבע. כלומר בהינתן פרמטר p, פונקציית פילוג ההסתברות:

$$P(x)=\begin{cases}p & ,x=1\\ 1-p, & x=0\end{cases}=p^x(1-p)^{1-x}$$

רגרסיה לוגיסטית

שימו לב שבניסוי הראשון במעבדה תדרשו לממש רגרסיה לוגיסטית בעצמכם.

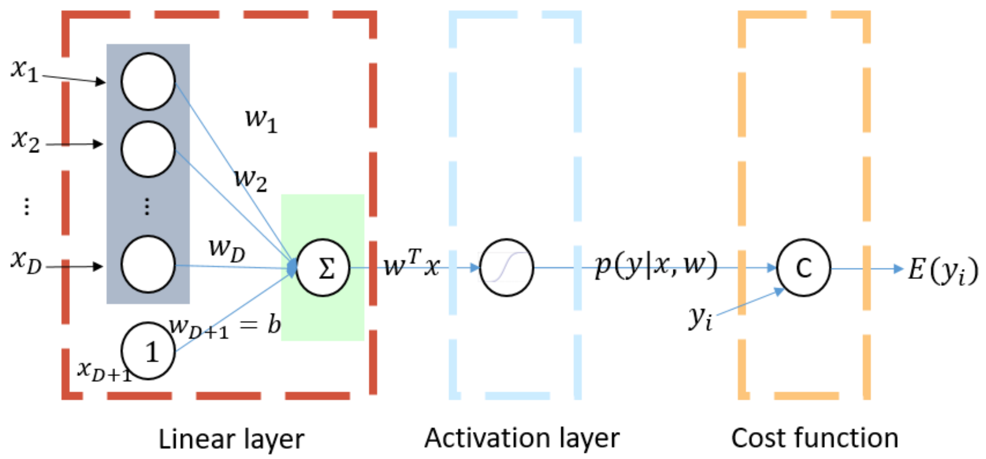
רגרסיה לוגיסטית הינו אלגוריתם סיווג בינארי שפותח בשנת 1958 ע״י David Cox.

אנחנו נתבונן בבעיה זו כעוד מקרה פרטי של רשת נוירונים.

השוני העקרוני העיקרי, הוא שמודל זה מנסה למדל את *ההסתברות* של דוגמה להיות ממחלקה מסוימת, ולא כקביעה חד משמעית (hard) שהיא שייכת למחלקה.

ניתן להסתכל על מודל זה כמורכב משלוש שכבות: 1. שכבה לינארית (אפינית). 2. שכבת אקטיבציה (שכבה לא-לינארית). 3. פונקציית מחיר Cross entropy.

גרפית, נשרטט את המודל כך:



איור 11 – איור סכמתי של מודל Logistic regression

כאמור, כעת מוצא המודל הינו $p(y \mid x, w)$.

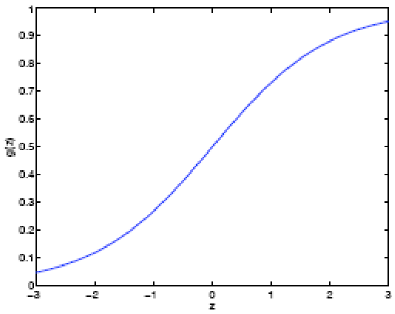
משיגים את הייצוג ההסתברותי ע"י שימוש בפונקציית הסיגמואיד :

$$p(y|x,w) = Ber\big(y\big|\sigma(w^T x)\big)$$

כאשר הסיגמואיד מוגדר כך :

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

וגרפית נראה כך :



איור 12 – גרף של פונקציית הסיגמואיד

כלומר, בהינתן וקטור משקולות w, ודוגמה x_i, הסיכוי של y_i להיות מחלקה מספר 1 מתפלג לפי ברנולי עם פרמטר $p = \sigma(w^T x)$.

*שימו לב שהפונקציה מוציאה ערכים בין 0 ל-1, כנדרש מפונקציית פילוג הסתברות.

בצורה מפורשת, הביטוי אותו מחשבים הינו :

$$P(y_i|x_i,w) = \begin{cases} \pi_{i0} \triangleq 1 - \dfrac{1}{1+e^{-w^T x}}, & y_i = 0 \\ \pi_{i1} \triangleq \dfrac{1}{1+e^{-w^T x}}, & y_i = 1 \end{cases}$$

כלומר, π_{i0} היא ההסתברות עבורה y_i יקבל ערך 0 ו-π_{i1} היא ההסתברות עבורה $\mathrm{y_i}$ יקבל ערך 1.

כאמור, נרצה למזער את ההסתברות לשגיאה על פני כל הדוגמאות :

$$P(Y|X,w) \overset{i.i.d}{=} \prod_{i=1}^{n} Ber\big(y_i\big|\sigma(x_i w)\big) \triangleq \prod_{i=1}^{n} \pi_{i0}^{\mathbb{1}_0(y_i)} \pi_{i1}^{\mathbb{1}_1(y_i)}$$

כאשר השתמשנו בהנחת ה-i.i.d. של הנתונים (התפלגות משותפת = מכפלת ההתפלגויות) ובהגדרת התפלגות ברנולי. כמו כן,

$$\mathbb{1}_0(y_i) = \begin{cases} 1, y_i = 0 \\ 0, else \end{cases} = 1 - y_i, \quad \mathbb{1}_1(y_i) = \begin{cases} 1, y_i = 1 \\ 0, else \end{cases} = y_i$$

הבאת ההסתברות הנ״ל למקסימום שקול למזעור מינוס הלוג של אותו הביטוי :

$$NLL(Y,X,W) = -\sum_{i=1}^{n} \mathbb{1}_0(y_i) log\pi_{i0} + \mathbb{1}_1(y_i)\log\pi_{i1} \left(= -\sum_{i=1}^{n}(1-y_i)log\pi_{i0} + y_i\log\pi_{i1}\right)$$

הביטוי הנ״ל נקרא מינוס לוג שגיאת ההסתברות (Negative Log Likelihood = NLL) או ה-Cross Entropy.

<u>חישוב הגרדיאנט</u>

במקרה של רגרסיה לוגיסטית, הפונקציה אותה נרצה למזער הינה:

$$f(w,x,y) = NLL(Y,X,W) = -\sum_{i=1}^{n}(1-y_i)log\pi_{i0} + y_i\log\pi_{i1}$$

$$= -\sum_{i=1}^{n}(1-y_i)log(1-\pi_{i1}) + y_i\log(\pi_{i1})$$

מהי הנגזרת של הביטוי הנ״ל?

כדי לחשב ביטוי מפורש, נשתמש בכלל השרשרת, כאשר נגדיר $z = w^T x$:

$$g(w,x_i,y_i) = \frac{\partial f}{\partial w} = \frac{\partial f}{\partial \pi_{i1}}\frac{\partial \pi_{i1}}{\partial z}\frac{\partial z}{\partial w}$$

כעת נרשום את הביטויים בצורה מפורשת :

$$\frac{\partial f}{\partial \pi_{i1}} = \frac{(1-y_i)}{1-\pi_{i1}} - \frac{y_i}{\pi_{i1}}$$

$$\frac{\partial \sigma}{\partial z} = \sigma(z)\big(1-\sigma(z)\big)$$

$$\frac{\partial z}{\partial w} = x_i$$

הוכחת הביטוי השני והשלישי הן שאלות מספר 3 ו-4 בתרגילי ההכנה.

שימו לב כי בשאלה 3 נבצע גזירה מטריצית השייכת לחדו״א מטריצית. כללי הגזירה המטריציים מאוד דומים לאלו של סקלרים. במעבדה זו לא ניכנס להוכחתם.

המעוניינים בפיתוח המלא, יכולים למצוא הסבר תמציתי על הנושא ניתן למצוא כאן.

אלגוריתם אימון רגרסיה לוגיסטית

קלט: סדרת דוגמאות מתויגות לאימון $\{x_k, y_k\}_{k=1}^{n}$, פרמטר $\alpha > 0$,

מספר חזרות מרבי על סדרת הלימוד K.

פלט: וקטור פרמטרים $w = (w_1, w_2, ..., w_d, w_{d+1})$.

שלבי הלימוד:

1. אתחול: אתחל את הווקטור w^0 בערך כלשהו (למשל, $w^0 = (1,1,...,1)$).
2. עבור על דוגמאות סדרת האימון $i = 1,2,3 \dots, n$:
   - 2.1. קבל את הווקטור x_i והתיוג $y_i \in \{0,1\}$.
   - 2.2. קבע את קצב הלימוד η_t.
   - 2.3. חשב את שגיאת המודל עם וקטור המשקלים הנוכחי w^0: $NLL(x, y, w)$.
   - 2.4. חשב את הווקטור $g(x, y, w)$.
   - 2.5. עדכן את w^t (שהוא **וקטור** המשקולות אשר התקבל לאחר איטרציה t):

$$w^{t+1} = w^t - \alpha g(w, y, x)$$

3. חזור על שלב 2 עד שמתקיים אחד משני תנאי העצירה הבאים:
   - 3.1. אין יותר עדכון של w - האלגוריתם מנבא נכון את התיוג של כל הדוגמאות.
   - 3.2. הושלמו K חזרות על סדרת הלימוד (epochs).

הערות:

- גם אלגוריתם זה הוא מקרה פרטי של האלגוריתם שראינו קודם לאימון רשתות.
- הערכים של y הפעם הם $\{0,1\}$ במקום $\{-1,1\}$. בפועל כל מספר מייצג מחלקה (לדוגמה: סוג דג או סוג של פרח).

רגרסיה לוגיסטית כמודל שכבות

כעת ננסה להכליל את שלבים 2.2 ו-2.3 למודל שכבות כללי תוך הסתכלות על רגרסיה לוגיסטית כמקרה בוחן. רשתות נוירונים מאמנים באמצעות שתי פונקציות רקורסיביות:

1. פונקציית מעבר קדמית (שלב 2.3).
2. פונקציית מעבר אחורית (שלב 2.4).

פונקציית המעבר הקדמית של המודל מוגדרת כהרכבה של השכבות:

$$f(x, y) = C\left(\sigma\big(L(x)\big)\right)$$

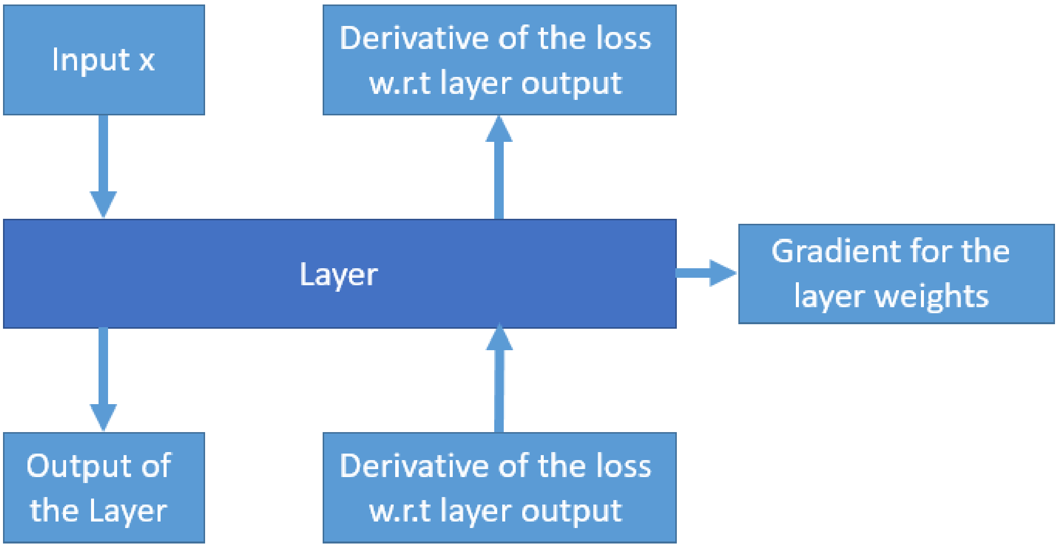
כאשר C היא פונקציית המחיר, σ היא הסיגמואיד ו-L היא הפונקציה האפינית.

פונקציית המעבר האחורית היא למעשה חישוב הנגזרות לכל שכבה. פעם לפי הקלט של השכבה, ופעם לפי הפרמטרים של השכבה אם קיימים:

$$g(w, x_i, y_i) = \frac{\partial f}{\partial w} = \frac{\partial f}{\partial \sigma}\frac{\partial \sigma}{\partial z}\frac{\partial z}{\partial w}$$

לאחר המעבר הקדמי והאחורי, ניתן לעדכן את משקולות (פרמטרי) המודל לפי gradient descent או כל מודל מתקדם אחר.

נוכל להסתכל על כל שכבה כיחידה מודולרית ועצמאית:



איור 13 – שכבה כללית ברשת נוירונים, כניסות ויציאות

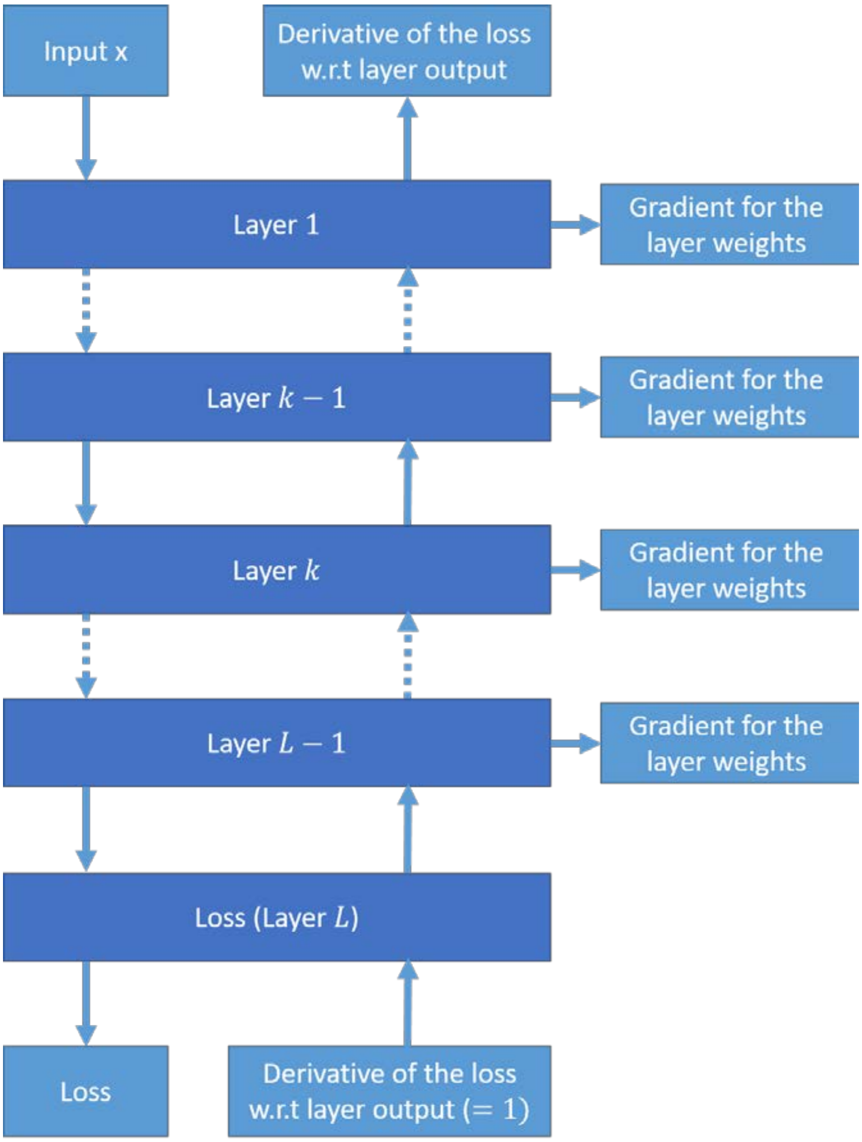
על מנת שנוכל להשתמש בה בלימוד הרשת, עליה לממש שלוש פונקציות:

1. פונקציית מעבר קדמית.
2. נגזרת לפי פרמטרי השכבה.
3. נגזרת לפי כניסת השכבה.

רשמו בצורה מפורשת את שלוש הפונקציות של השכבות ברגרסיה לוגיסטית עבור המקרה הסקלרי:

	מעבר קדמי	נגזרת לפי פרמטרים	נגזרת לפי הכניסה
שכבה ליניארית			
סיגמואיד		-	
NLL		-	

לאחר שחישבנו (ומימשנו) את שלוש הפונקציות לכל שכבה, ניתן לחבר את השכבות יחדיו ולאמן את כל המודל באמצעות gradient descent. שיטת אימון זו מכונה בשם backpropagation. למעשה, מדובר בלא יותר מאשר הפעלת כלל השרשרת.



איור 14 – רשת נוירונים כמודל רב שכבתי

בעיית סיווג מתוך מחלקות רבות

עד כה דנו בבעיית סיווג בינארית- בהינתן נקודה במרחב עלינו להחליט לאיזו מחלקה מבין שתיים נתונות הנקודה שייכת. במקרה כזה, ברשת הנוירונים שאנו בונים מספיקה יציאה יחידה אשר תקבע את השתייכות הכניסה (1 או 0).

כאשר אנו צריכים לקבוע שיוך לאחת מתוך $N>2$ מחלקות, וקטור הסיווג הנתון יהיה וקטור בגודל N ויכיל אפסים בכל המיקומים פרט לאחד - באינדקס המציין את המחלקה אליה משתייכת הנקודה הרלוונטית. נרצה שמוצא הרשת יהיה וקטור בגודל N המכיל עבור כל אינדקס את ההסתברות של

השתייכות הכניסה למחלקה באינדקס זה. כלומר, וקטור של N ערכים בין 0 ל-1, כאשר סכום הערכים הוא 1, והערך הגבוה ביותר יתאים למחלקה אשר ההשתייכות אליה תהיה בעלת הסבירות הגבוהה ביותר. על מנת להגיע למוצא כנדרש, נשתמש בפונקציית אקטיבציה הנקראת softmax. נסמן את מוצא השכבה הקודמת ל-softmax ע״י o.

$$softmax(o_i) = \frac{e^{o_i}}{\sum_{k=1}^{N} e^{o_k}}$$

שימו לב כי עבור כל מחלקה נקבל כנדרש, ערך בין 0 ל-1 כאשר סכום על הערכים על המחלקות הוא 1. גם עבור פונקציה זו, נחשב את הגרדיאנט עבור ה-backpropagation.

נגזור את פונקציית ה-softmax לפי הכניסה, כלומר נחשב $\frac{\partial softmax(o)}{\partial x_i}$. נסמן $p_i = softmax(o_i)$ ונפריד למקרים:

$$\frac{\partial softmax(o_i)}{\partial o_i}, \qquad \frac{\partial softmax(o_{j\neq i})}{\partial o_i}$$

בתרגיל ההכנה תראו כי מתקיים:

$$\frac{\partial p_i}{\partial o_j} = \begin{cases} p_i(1-p_i), & i = j \\ -p_i p_j\,, & i \neq j \end{cases} = p_i(\delta_{ij} - p_j) \qquad\qquad \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

על מנת לפשט את הביטוי, נגזור את שכבת ה-softmax יחד עם שכבת הNLL הסופית ברשת.

עבור בחירה מתוך N מחלקות, פונקציית ה-NLL נראית כך:

$$NLL(Y, X, W) = -\sum_{i=1}^{N} y_i \log p_i$$

$$\frac{\partial NLL}{\partial p_i} = -\frac{y_i}{p_i}$$

נגזור את הפונקציה NLL ביחס למשתנה הכניסה של ה-softmax, o_j:

$$\frac{\partial NLL}{\partial o_j} = -\sum_{i=1}^{N} y_i \frac{\partial \log(p_i)}{\partial o_j} = -\sum_{i=1}^{N} \frac{y_i}{p_i}\frac{\partial p_i}{\partial o_j} = -\sum_{i=1}^{N} \frac{y_i}{p_i} p_i(\delta_{ij} - p_j) = -\sum_{i=1}^{N} y_i(\delta_{ij} - p_j)$$

$$= -y_j + p_j \underbrace{\sum_{i=1}^{N} y_i}_{=1} = p_j - y_j$$

לקריאה נוספת

[1] *Pattern Classification* (2nd ed.), Richard O. Duda, Peter E. Hart and David G. Stork (John Wiley and Sons, 2001)

[2] *Deep Learning*, Ian Goodfellow and Yoshua Bengio and Aaron Courville (MIT Press, 2016)

מפגש ראשון

שאלות הכנה

1. עבור כל אחת מן הבעיות הבאות קבעו: האם מדובר בבעיית סיווג או רגרסיה, הציעו מרחב דוגמאות ומרחב תיוג.
   א. זיהוי כתב יד מתמונה
   ב. מסנן דואר זבל (Spam)
   ג. זיהוי דובר מקטע אודיו
   ד. חיזוי שער מניה לאחר מספר ימים
   ה. חיזוי משך נסיעה בין שני יעדים

   הציעו עוד שתי דוגמאות לבעיות משלכם ונתחו אותן באותו האופן.

2. עבור כל אחת מהשכבות הבאות (כל אחת בנפרד), פתחו את שלוש הפונקציות (forward –הפונקציה עצמה, ושני ה-backward- גזירה של הפונקציה לפי הקלט ולפי הפרמטר):

1. $\mathrm{f(x,a)} = \log(\mathrm{ax})$
2. $MSE(x,y) = (x-y)^2$
3. $\mathrm{f(x)} = \max(0,\mathrm{x})$
4. $\mathrm{f(x,a,b)} = \sqrt{\mathrm{ax+b}}$

*ניתן להניח שהפונקציות והמשתנים הינם סקלריים.

3. עבור הפונקציה המטריצית הבאה $f: \mathbb{R}^3 \to \mathbb{R}^2$:

$$f(X) = A^T X = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

כאשר $X \in \mathbb{R}^3$ ו-$A \in \mathbb{R}^{3\times2}$.

פתחו את פונקציות ה-backward של השכבה.

שימו לב שהגרדיאנט של f נתון ע"י:

$$\nabla_X f(X) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \end{bmatrix}$$

הדרכה: כדי לחשב את ה-backward של השכבה ביחס לפרמטרים נגדיר את השכבה בצורה הבאה:

$$f(X) = \begin{bmatrix} f_1(X) \\ f_2(X) \end{bmatrix} = \begin{bmatrix} A_1^T X \\ A_2^T X \end{bmatrix}$$

כאשר:

$$A_1 = \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \end{bmatrix}, A_2 = \begin{bmatrix} a_{12} \\ a_{22} \\ a_{32} \end{bmatrix}$$

הנגזרת של f לפי A מוגדרת כך:

$$\nabla_A f = \begin{bmatrix} \nabla_{A_1} f_1 \\ \nabla_{A_2} f_2 \end{bmatrix}$$

4. יש להראות כי הנגזרת של הפונקציה $\sigma(z)$ נתונה ע״י הביטוי:

$$\frac{\partial \sigma}{\partial z} = \sigma(z)\big(1 - \sigma(z)\big)$$

5. *Gradient descent*

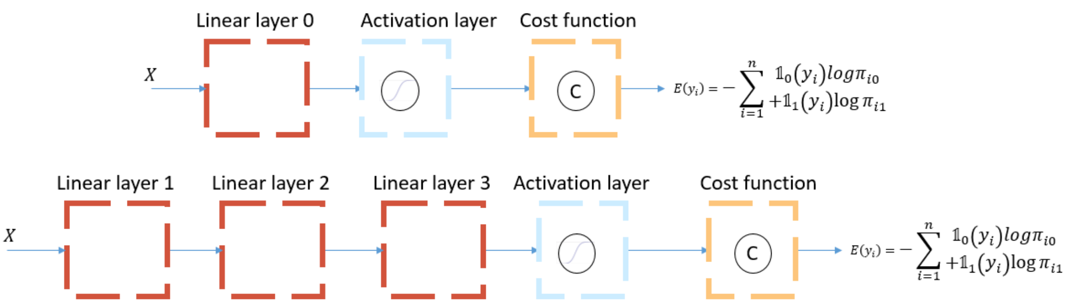
בהינתן הפונקציה $f(x) = x^2$ ונקודת התחלה $x_0 = -1$, יש להדגים שלוש איטרציות של gradient descent.

א. עבור גודל צעד $\eta = 0.01$.

ב. עבור גודל צעד $\eta = 2$.

מהי חשיבותו של בחירת גודל הצעד בתהליך הלימוד?

6. האם הרשתות הבאות שקולות? הסבירו את תשובתכם.



7. צרפו לדו״ח כתמונה (ברורה) את החלקים שהשלמתם מתוך כל הקוד עבור סיווג האירוסים.

8. הגישו את הטבלה המלאה *מרגרסיה לוגיסטית כמודל שכבות* בה יש למלא את הפונקציות של כל שכבה במודל. שימו לב שבטבלה זו כל שכבה עומדת בפני עצמה ואינה תלויה באחרות.

9. עבור פונקציית ה-softmax, פתחו את הביטויים לנגזרות והראו שמתקיים:

$$\frac{\partial p_i}{\partial o_j} = \begin{cases} p_i(1-p_i), & i = j \\ -p_i p_j\,, & i \neq j \end{cases} = p_i\big(\delta_{ij} - p_j\big) \qquad \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

מפגש ראשון - מהלך הניסוי

בניסוי זה נממש רגרסיה לוגיסטית עבור בעיית סיווג האירוסים בדומה לנעשה בתרגיל ההכנה. לאחר מכן נתנסה ברשתות עמוקות הכוללות מספר גדול יותר של שכבות לסיווג ספרות מתוך תמונות. לבסוף נכיר את ה-toolbox המובנה של MATLAB לבניה ואימון של רשתות.

הנחיות

שאלות הדו"ח המסכם מופיעות לאורך הניסוי. יש לענות עליהן בכיתה, במהלך ביצוע הניסוי.
בכל פעם שתצטרכו לבצע משימה כלשהי במהלך הניסוי, המשימה תופיע ליד סימן קבוע. לדוגמה:

- ❖ הריצו את הקובץ deepLearningIsFun.m (דוגמה של סימון משימה בקובץ מהלך הניסוי)

השאירו את קבצי הקוד בתיקיות כפי שקיבלתם אותם (אין צורך להוציא את כולם לאותה תיקיה).

בנוסף, עליכם לצרף לדו"ח המסכם את כל קבצי ה-MATLAB (קוד, תוצאות וגרפים) שיצרתם במהלך הניסוי.

דרישות מחשוב:

- MATLAB 2018a לפחות
  - Neural Network Toolbox
  - Parallel Computing Toolbox
  - Statistics and Machine Learning Toolbox
  - AlexNet pre-trained network

חלק א' – רגרסיה לוגיסטית

בשלב הראשון בניסוי, נממש רגרסיה לוגיסטית עבור בעיית סיווג האירוסים.

נזכיר, כי רשתות נוירונים מאמנים באמצעות שתי פונקציות רקורסיביות:

1. פונקציית מעבר קדמית (שלב 2.3 באלגוריתם).
2. פונקציית מעבר אחורית (שלב 2.4 באלגוריתם).

ניעזר בטבלה שחישבנו בתרגיל הכנה מספר 8 על מנת לממש רשת בשלבים.

בדומה לתרגיל ההכנה, מצורפים קטעי קוד MATLAB חלקיים אשר ממשים מודולים המרכיבים מערכת היררכית של רגרסיה לוגיסטית:

- logistic_regression.m
  - forward functions:
    - affine_forward.m
    - nll_forward.m
    - sigmoid_forward.m
  - backward functions:
    - affine_backward.m
    - nll_backward.m
    - sigmoid_backward.m

1. עבור רגרסיה לוגיסטית כמודל שכבות- בהינתן וקטור כניסה לרשת $x \in \mathbb{R}^d$ ווקטור מוצא $y \in \mathbb{R}$, מהו הממד של w וקטור המשקולות הכולל בתוכו את רכיב ה-bias?

❖ הסתכלו בקבצי ה-MATLAB הנתונים לכם ופתחו את התיקייה lab1.

2. הסתכלו על הפונקציות הנתונות בתיקייה layers. מהו סדר השימוש הנכון בפונקציות עבור תהליך ה-forward ועבור תהליך ה-backward?
   - שימו לב כי כל אחת מהפונקציות מתאימה לפונקציה בטבלה שהגשתם בתרגיל ההכנה.
   a. מהו הממד הצפוי למוצא כל אחת מהפונקציות הללו (עבור וקטור כניסה $x \in \mathbb{R}^d$)?

❖ השלימו את הקוד והריצו טסטים עבור הפונקציות affine_forward, nll_forward, sigmoid_forward, affine_backward, nll_backward, sigmoid_backward וודאו כי אכן מתאימים הממדים להם ציפיתם בסעיף הקודם.

דגשים חשובים שצריך לקרוא לפני שמתחילים לממש:

- כל פונקציית backward צריכה להחזיר את הגרדיאנט המלא הנוכחי, כלומר הגרדיאנט המחושב בצעד הנוכחי כפול הגרדיאנט שחושב עד כה (לפי כלל השרשרת).
- על המימוש שלכם עבור כל פונקציה לתמוך בכניסה יחידה (x שהוא וקטור עמודה) וגם בכניסה שהיא שרשור של מספר כניסות (x שהוא מטריצה, שרשור של מספר וקטורי עמודה).
- **השתמשו בקבצי הבדיקה המצורפים בתיקייה layers/tests:**
  - הריצו את הקובץ run_tests.m.
  - ודאו כי הבדיקות עברו. בפלט ההרצה תוכלו לראות פירוט של הטסטים המורצים ואת הבדיקות אשר עברו או נכשלו.
  - במידה וקיימות בדיקות שנכשלו, תקנו את הפונקציות הרלוונטיות לבדיקה וחזרו על הבדיקה.

❖ השלימו את הקוד עבור הפונקציה logistic_regression.

3. מהי הצורה שבה אתם מצפים שיהיה קו ההפרדה של הרגרסיה הלוגיסטית?

❖ הריצו את iris_logistic_regression פעמיים, פעם אחת עם קצב התכנסות 0.1 ופעם שניה עם קצב התכנסות 0.01. צפו בתהליך ההתכנסות. הוסיפו לדוח המסכם שלכם את פלטי הריצות.

4. ציינו את ההבדלים בין שתי ההרצות שביצעתם והסבירו- מה גרם להבדלים אלו?

חלק ב' – זיהוי ספרות בתמונות (סיווג בעזרת רשת נוירונים)

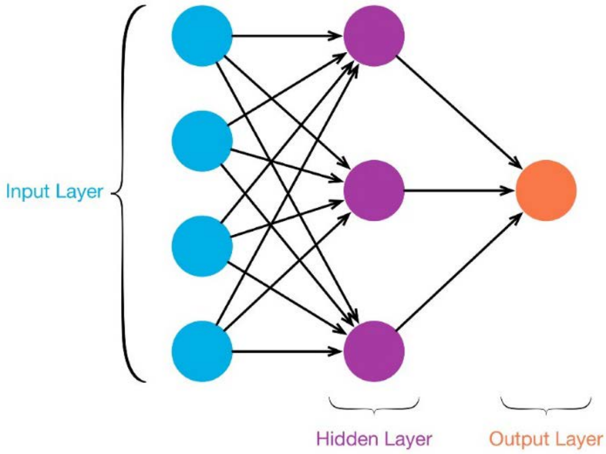
כעת נעבור לבעיה יותר מורכבת: זיהוי ספרות כתב יד.

מאגר המידע בו נשתמש הוא MNIST, המכיל תמונות של ספרות, ותיוגים מתאימים של הספרה המתאימה לכל תמונה. כלומר, הקלט לרשת במקרה זה יהיה תמונה, והפלט הרצוי יהיה הקטגוריה המתאימה לספרה בתמונה (Classification).

❖ פתחו את הקובץ applyTwoLayerPerceptronMNIST והריצו את המקטע הראשון (mnistDataPrep). התבוננו בתמונות לדוגמה, ובחלוקת התיוגים.

הרשת שאנו נבנה תהיה בעלת *שתי שכבות*. כלומר, כל שכבה מכילה הכפלה במטריצה (ליניארית), ואקטיבציה. בהתאמה, לרשת תהיה שכבה נסתרת אחת של נוירונים.

כלומר, הרשת תיראה כך (אילוסטרציה בלבד, הגדלים אצלכם כמובן שונים):



שימו לב שאת גודל השכבה הנסתרת ניתן לבחור כרצוננו.
כל שכבה ליניארית באה לידי ביטוי ע"י טור של חצים השחורים, והאקטיבציה מתבצעת בכל נוירון (עיגול סגול או כתום).
שכבת האקטיבציה הראשונה בה נשתמש היא סיגמויד, בשכבה השנייה נשתמש באקטיבציה מסוג softmax ופונקציית ההפסד היא Negative Log Likelihood.

1. רשמו את הביטויים המתמטיים עבור שלבי תהליך ה-forward של רשת כזו. התייחסו לכל שלבי הביניים (אחרי כל שכבה ליניארית/אקטיבציה/loss).
השתמשו בסימונים הבאים:
   a. x וקטור הכניסה לרשת
   b. y וקטור המוצא של הרשת
   c. w_1, w_2 המשקולות בשכבה הראשונה והשנייה בהתאמה
   d. b_1, b_2 רכיב ה-bias בשכבה הראשונה והשנייה בהתאמה
   e. $\sigma(x)$ פונקציית האקטיבציה סיגמויד
   f. $softmax(x)$ פונקציית האקטיבציה softmax
   g. $l(\hat{y}, y)$ פונקציית ההפסד

   איך גודל השכבה הנסתרת משפיע על הביטויים המתמטיים?

כפי שראינו, כדי לאמן את המודל, יש לחשב את שלושת הפונקציות עבור כל שכבה.
למעשה, כבר חישבנו כמעט את כל השכבות והנגזרות הנחוצות למודל כשביצענו את הרגרסיה הלוגיסטית.
נשים לב כי במקרה של MNIST, בניגוד למשימה הקודמת אנו בוחרים מחלקה אחת מתוך 10 מחלקות ולא מבצעים סיווג בינארי. לכן, יש להתאים את פונקציית הNLL.

- ❖ הסתכלו בתיקייה המתאימה למשימה זו. פתחו את התיקייה layers-mnist, והשלימו את הקוד עבור הפונקציות הבאות לפי הנוסחאות המוצגות בחומר הרקע למעבדה:

- nll_forward_mnist.m
- softmax_forward.m
- nll_and_softmax_backward.m

גם כאן תקפים אותם הדגשים מחלק א׳. **השתמשו בטסטים** הניתנים לכם על מנת לבדוק את הפונקציות שכתבתם.

שימו לב כי כלל השרשרת פועל גם **באופן וקטורי**. לכן, שימו לב לממדי ההכפלות ולשימוש בכפל איבר-איבר (.*) מול כפל מטריצי (*) וכנ״ל גם בחלוקה.

בקטעי הקוד מצורפים לכם קבצים המממשים רשת זו. הקבצים מחולקים באופן הבא :

- applyTwoLayerPerceptronMNIST- מעטפת כללית אשר מכינה את מאגר המידע, שולחת אותו לאימון ולבסוף שולחת לוולידציה ומחשבת error.
- forward_pass – מבצעת מעבר קדמי על הרשת.
- backward_pass – מבצעת back-propagation על הרשת.
- trainTwoLayerPerceptron- מאמנת את הרשת שלנו ומחזירה את המשקולות לאחר האימון.
- validateTwoLayerPerceptron- מבצעת forward pass על סט הוולידציה ומחזירה שגיאה.

- ❖ השלימו את השורות הנחוצות בtrainTwoLayerPerceptron.
- ❖ השלימו את הקוד ב-forward_pass וב-backward_pass על ידי השכבות שכתבתם בעצמכם בחלק א׳ ובחלק ב׳.
- ❖ הריצו את applyTwoLayerPerceptronMNIST. ודאו כי המודל מתכנס, וצרפו את התוצאות הסופיות והגרף המתקבל.

2. מהו גודל השכבה הנסתרת כרגע בקוד? הסבירו כיצד גודל השכבה הנסתרת משפיעה על המודל. התייחסו בתשובתכם לסיבוכיות המודל ושיקולי ביצועים (מקום, זמן, דיוק). (הציצו באיור 5 ואיור 6 לקבלת אינטואיציה)

חלק ג׳ – סיווג ספרות בעזרת Matlab Neural Network Toolbox

שימו לב- חלק זה אינו תלוי בחלקים הקודמים במעבדה זו.

כעת, נבנה רשת זהה לרשת אשר בנינו בחלק ב׳ בעזרת כלי רשתות הנוירונים של MATLAB.

מאגר המידע בו נשתמש הוא מאגר דומה מאוד לMNIST, המכיל תמונות ותיוגים שהם הספרות בתמונות.

- ❖ קראו את נספח א׳ – Matlab Layers חלק א׳, והבינו את מטרתה של כל שכבה.

בניגוד לחלק הקודם, האקטיבציה הראשונה בה נשתמש היא ReLU, ולא סיגמויד. בחלק ההכנה למפגש הבא נלמד מה ההבדל ביניהם ומה היתרונות של ReLU. השימוש בשכבת ה-ReLU זהה לשימוש שעשינו בסיגמויד. האקטיבציה השנייה שנשתמש בה תהיה softmax.

❖ בקובץ mnistClassificationDL, השלימו את השכבות הרצויות לפי המודל מהסעיף הקודם עם השינויים שצוינו בפסקה הקודמת.
עליכם להחליף כל שורה המתארת שכבה בשכבה של מטלב המתאימה לשורה זו.
רשימת השכבות הרלוונטיות לניסוי זה נמצאת בנספח א׳ – Matlab Layers חלק א׳. שימו לב לפרמטרים של כל שכבה.

1. הסבירו מה המשמעות של כל אחד מהפרמטרים המתקבלים כקלט לפונקציה trainingOptions בקובץ mnistClassificationDL.

❖ הריצו את mnistClassificationDL וצפו בתהליך ההתכנסות של הרשת. שימו לב לתדפיס בcommand window של MATLAB :

Epoch	Iteration	Time Elapsed (seconds)	Mini-batch Loss	Validation Loss	Mini-batch Accuracy	Validation Accuracy	Base Learning Rate

הפרמטרים אשר יעניינו אותנו בעיקר בחלק זה הם ה-Loss וה-Accuracy, של ה-train (*רשום כMini-batch*) למול ה-Validation.
ניתן לראות את ה-Loss וה-Accuracy של סט האימון בגרף ההתכנסות אשר מוצג בתהליך הלימוד. שימו לב שמוצגים בקו מקווקו נתוני ה-validation.

❖ הריצו את המודל במשך 10 epoch-ים עם גודל צעד 0.0001. הסתכלו על התכנסות ה-Loss לאורך האימון.
❖ הדביקו את גרף ההתכנסות ואת הטבלה מה-command. רשמו את ה-Loss וה-accuracy הסופיים של ה-train וה-validation.

2. מה הסיבה להבדלים ב-Loss בין ה-train וה-validation? איזו תופעה מתרחשת כאן? איך לדעתכם ניתן לפתור את התופעה הזו?

במפגש השני של המעבדה נראה מספר דרכים להתמודד עם תופעה זו ונדון בהן רבות. בסעיפים הבאים אתם יכולים להתעלם מתופעה זו בתשובותיכם.

❖ כעת, הריצו את המודל עם גודל צעד 0.5.
הדביקו את גרף ההתכנסות ואת הטבלה מה-command. רשמו את ה-Loss וה-accuracy הסופיים של ה-train וה-validation.

3. מה קרה לגרף ההתכנסות? הסבירו מדוע.

❖ מצאו את גודל הצעד הגדול ביותר עבורו הרשת מצליחה להתכנס תוך 10 epochs (דיוק של 3 ספרות אחרי הנקודה בגודל הצעד), כלומר מצאו גודל צעד עבורו ניתן לראות מגמה מתאימה עבור ה-loss וה-accuracy.

4. השוו בין תהליך ההתכנסות והתוצאות הסופיות עם גודל צעד 0.0001,0.001. מה ההבדלים ביניהם? למה שינוי גודל הצעד הביא להבדלים אלו? התייחסו לנתונים בטבלה ולצורת הגרף.
5. מה הייתם מצפים שיקרה עבור גודל צעד 0.00001? (מוזמנים להריץ ולבדוק את תשובתכם)
   a. מהו גודל הצעד המתאים ביותר מבין שלושת האופציות 0.001,0.0001,0.00001? האם גודל זה אידיאלי לאורך כל תהליך ההתכנסות?
   b. האם גודל זה יתאים לכל רשת נוירונים שנרצה לאמן? מדוע?

רקע למעבדה – מפגש שני

הקדמה

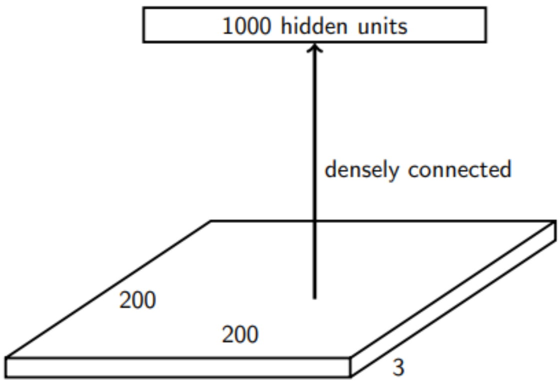
בחלק השני של הניסוי נכיר שכבות נוספות המהוות אבני בסיס בלמידה עמוקה. בפרט, נכיר שכבה הנקראת "שכבת קונבולוציה". לשכבה זו חשיבות גדולה בלמידה עמוקה והיא אפשרה לתחום לגדול לאן שהוא נמצא היום. כמו כן, נלמד על שכבות לא לינאריות, גישות אופטימיזציה מתקדמות, רגולריזציה ושיקולים בתכנון רשתות.

מבוא

במעבדה הקודמת השתמשנו בתמונות מאוד קטנות – 28x28x1. 1 מייצג את העובדה שמדובר בתמונות רמות אפור. בתמונות צבע יש 3 שכבות צבע.

בתחומים רבים נרצה לעבוד עם תמונות יותר גדולות. למשל תמונות בגודל 200x200x3. נציין שתמונות אלו קטנות מאוד בסטנדרטים של ימינו. יש בהן רק 40,000 פיקסלים, לעומת מצלמות פלאפון סטנדרטיות שמצלמות תמונות עם מיליוני פיקסלים.

רשת נוירונים שמקבלת תמונה כזו כקלט עשויה להיראות כך:



מה הבעיה עם שכבה כזו?

יש בה יותר מידי פרמטרים! לשכבה הראשונה (בלבד) יש $200x200x3x1000 = 120\ million$. זו כמות גדולה מאוד של פרמטרים. הן מבחינת המקום שהיא צורכת בזיכרון (32 ביט לכל פרמטר) והן מבחינת זמן האימון שיידרש עד שכל הפרמטרים יותאמו לדוגמאות האימון.

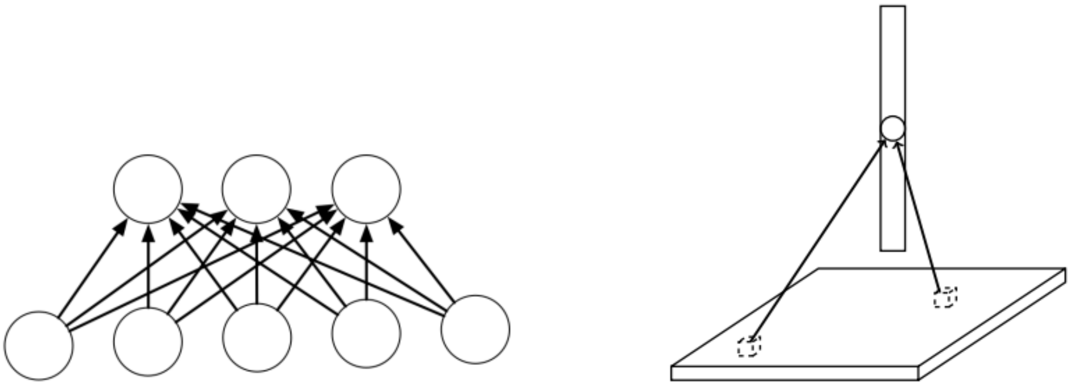
בעיה נוספת: מה יקרה אם נצלם את (כמעט) אותה התמונה רק בהיסט קטן ימינה? שימו לב שבמקרה זה כל פיקסל יוכפל בפרמטר אחר!

היינו רוצים לנצל תכונה של תמונות טבעיות שהיא שזהות של אובייקט לא תלויה במיקומו בתמונה. חתול בצד התמונה הוא חתול, כמו חתול הנמצא במרכז התמונה. תכונה זו נקראת אינווריאנטיות להזזה.

כעת נבנה בהדרגה את רשת הקונבולוציה, שפותרת את בעיות אלו.

רשתות קונבולוציה

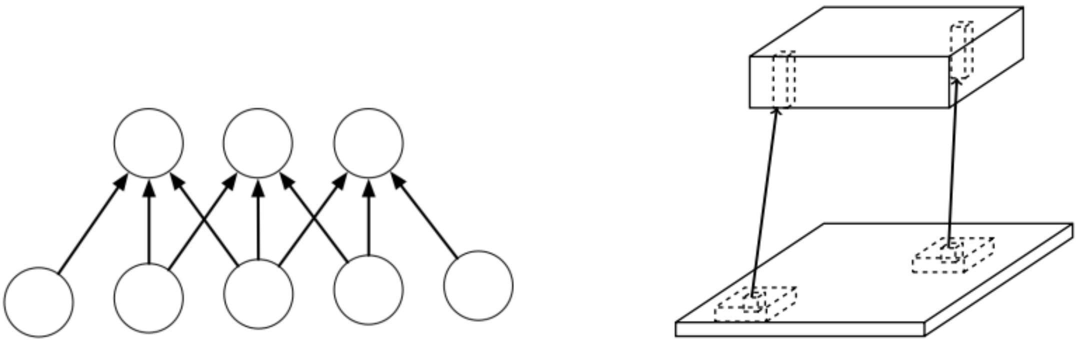
סכמתית נתאר רשתות לינאריות כך :



כל איבר במוצא הוא מכפלה של כל הפיקסלים בתמונה בסט המשקולות שלו. ניתן גם לחשוב על כל איבר במוצא כקומבינציה לינארית של כל הפיקסלים שבתמונה.

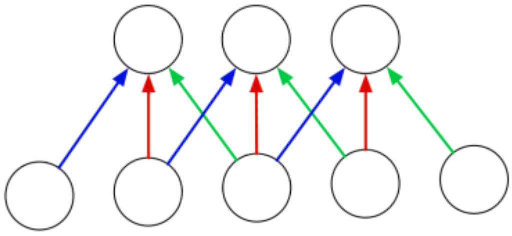
שכבות לינאריות לעיתים מכונות גם שכבות fully-connected, כלומר בעלות חיבוריות מלאה.

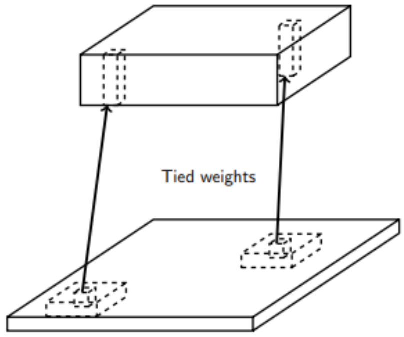
כעת נשנה מעט את השכבה. נרצה שמוצא השכבה תהיה גם היא מטריצה תלת-ממדית (גובה, רוחב ועומק. העומק במקרה של תמונת צבע הוא 3). כמו כן, נדרוש שכל פיקסל יושפע רק מסביבה קטנה בתמונת הכניסה. השכבה הנ״ל תיראה כך :



שכבה כזו מכונה שכבת locally connected, או בעלת חיבוריות מקומית. שימו לב שכל פיקסל במוצא מיוצג ע״י וקטור, בו כל איבר הוא קומבינציה לינארית **שונה** של האיברים בסביבתו (גובה, רוחב ועומק) בתמונת הכניסה.

כעת נעשה שינוי נוסף : המשקולות בכל המיקומים השונים יהיו שווים :



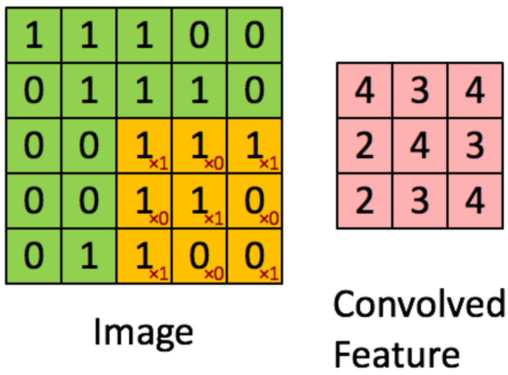


כלומר כל אזור בתמונת המוצא הוא תוצאה של מכפלות של אותן משקולות עם אזורים שונים בתמונת הכניסה. כל אוסף כזה של משקולות שמועבר על כל התמונה נקרא *גרעין קונבולוציה* או בקיצור *גרעין*.
פעולה זו היא בדיוק פעולת הקונבולוציה שראינו בקורס "אותות ומערכות" רק בדו-מימד.[4]
מתמטית, הביטוי נראה כך :

$$(f \star g)[n] = \sum_{i=0}^{m}\sum_{j=0}^{k} f[i][i+j]$$

<u>דוגמה לקונבולוציה :</u>

נראה כעת איך נראית פעולת הקונבולוציה עבור תמונה בגודל 5x5x1 עם ערכים בינאריים.[5]
ערכי התמונה הן הערכים שמופיעים בגדול עם רקע ירוק\צהוב. ערכי המשקולות מופיעים בקטן בצבע אדום עם רקע צהוב. תוצר פעולת הקונבולוציה היא התמונה שמופיעה מימין בוורוד.
שימו לב שבאזור עם הרקע הצהוב, אם נכפול את כל ערכי הפיקסלים בערכי המשקולות ונסכום, נקבל את הספרה 4 כפי שמופיע בתמונה הימנית בפינה הימנית התחתונה.



כפי שראינו קודם, תמונות יכולות להיות מורכבות מכמה שכבות. כיצד נראית פעולת הקונבולוציה אז?

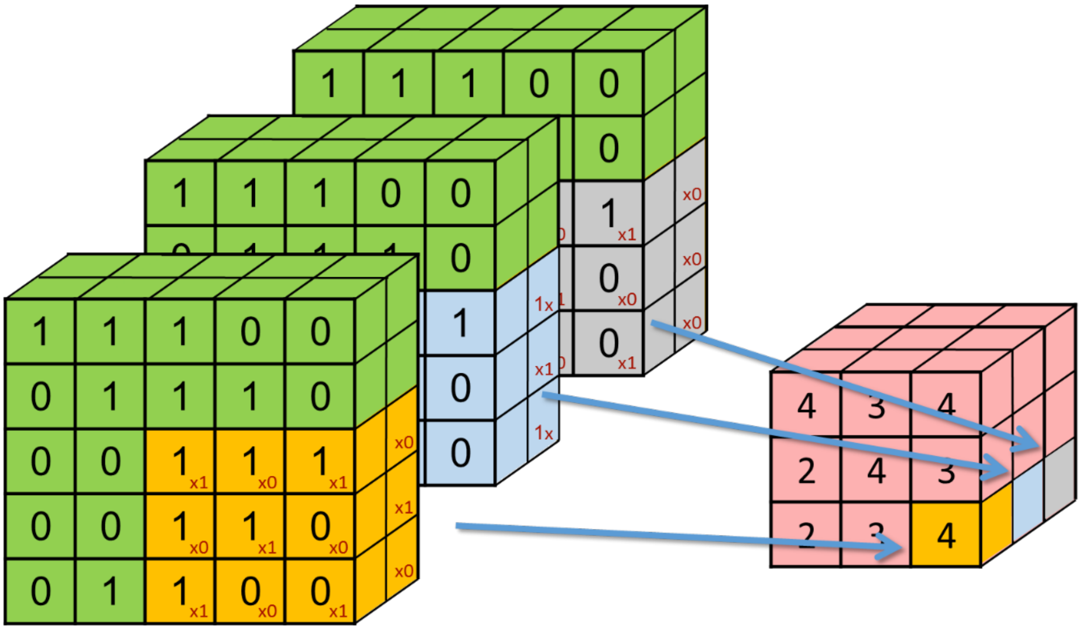
[4] זו היא כמעט אותה פעולה. בספרות של עיבוד אותות נהוג להגדיר את הקונבולוציה באמצעות שיקוף של אחת הפונקציות. הפעולה אותה אנו מציגים כאן היא ללא ההיפוך ונקראת פעולת *קרוס-קורלציה*. אנחנו ניצמד למונח "קונבולוציה" מכיוון שזהו המונח המקובל בתחום.

[5] חשוב להדגיש שרשתות נוירונים פועלות עם ערכים ממשיים ולא מספרים שלמים או בינאריים.

במקרה זה כל גרעין קונבולוציה הוא בעל גובה ורוחב (3x3 כמו קודם) אך בעל עומק התלוי בעומק תמונת הקלט. בדוגמה למטה תמונת הכניסה היא בעומק 2, לכן כל גרעין גם הוא בעומק 2. עומק תמונת המוצא יהיה כמספר גרעיני הקונבולוציה. בדוגמה למטה השתמשנו ב-3 גרעינים לכן גודל תמונת המוצא הוא 3.

הערה 1 - על אף שפעולות הקונבולוציה מבוצעות על מטריצות תלת-ממדיות (המכונות טנזורים), פעולת הקונבולוציה היא **דו-ממדית** שכן תנועת המסננים היא רק בשני צירים.

הערה 2 – בדומה לשכבות לינאריות, גם כאן נהוג להוסיף גודל קבוע לכל גרעין המכונה bias.



עד כה, ראינו שלושה פרמטרים שיש לבחור כשמגדירים שכבת קונבולוציה: אורך, רוחב ומספר גרעינים. נציג כעת עוד שני פרמטרים רלוונטיים.

ריפוד (padding) – בדוגמה לעיל רוחב וגובה התמונה קטנו כפונקציה של גודל התמונה המקורית וגודל גרעין הקונבולוציה. את הנוסחה לחישוב הגודל במוצא יש לחשב בתרגיל ההכנה מספר 1.

ניתן לשנות את רוחב וגובה תמונת המוצא ע"י ריפוד תמונת הכניסה באפסים.

דילוג (stride) – בדוגמה לעיל, עברנו על תמונת הכניסה עם הפילטרים כאשר כל פיקסל בתמונת המוצא התקבל ע"י הזזה של גרעין הקונבולוציה ב"צעד" אחד. ניתן גם לבצע צעדים גדולים יותר ולא לבצע קונבולוציה "מלאה".

הדגמה ויזואלית של ריפוד ודילוג ניתן לראות בקישור הזה.

שכבות הורדת מימד

שכבות נפוצות נוספות ברשתות הן שכבות הורדת מימד הנקראות pooling. שכבות אלה פועלות בצורה מקומית, בדומה לשכבות קונבולוציה.

מבין שכבות ה-pooling, הנפוצות ביותר הן שכבות max pooling ו- average pooling. פעולת השכבה היא הפעלת מקסימום וממוצע על קבוצת תאים בשכבה בהתאמה.

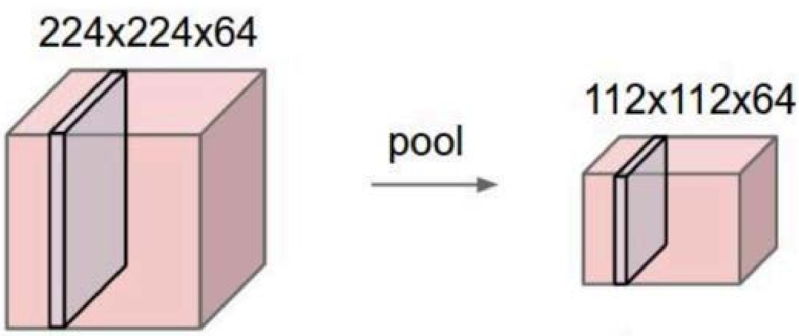
לשכבה זו 4 היפר-פרמטרים (פרמטרים שנקבעים מראש ולא נלמדים) ולא כוללת פרמטרים נלמדים כלל.

הפרמטרים הם הגודל של הפילטר (אורך ורוחב) וגודל הצעד (stride) בכל כיוון.

דוגמה מספרית לשכבה max pooling מומחשת באיור 15 – שכבת Max Pooling.

איור 15 – שכבת Max Pooling

כאשר מפעילים את השכבה על טנזור תלת מימדי, מפעילים את הפעולה על כל שכבה בנפרד. ראו איור 16 :



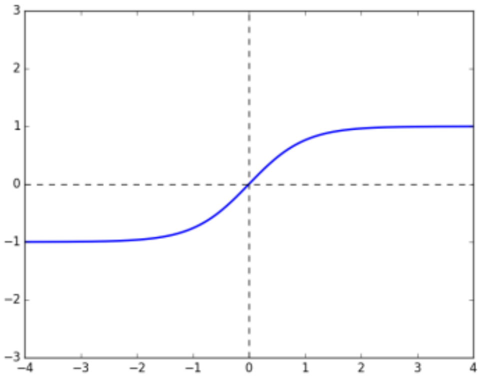
איור 16 – שכבת pooling על טנזור תלת מימדי

סוגי שכבות לא ליניאריות

עד כה ראינו שכבה לא לינארית אחת מסוג סיגמואיד. כעת נציג סוגים נוספים.

שכבה לא ליניארית אחרת שהייתה מאוד נפוצה הינה tanh:

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



שימו לב כי מתקיים הקשר הבא:

$$\tanh(x) = 2\sigma(2x) - 1$$

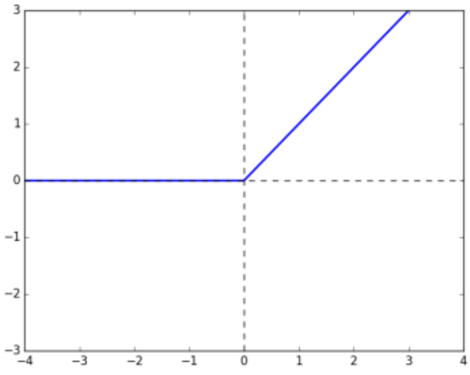
כלומר, $tanh$ היא לא יותר ממתיחה והזזה של פונקציית הסיגמאויד.

שתי שכבות לא ליניאריות אלו סובלות מבעיה הנקראת "בעיית הגרדיאנטים הדועכים" (vanishing gradients). הערך המקסימלי של פונקציית הנגזרת של סיגמואיד מתקבל עבור $x = 0$:

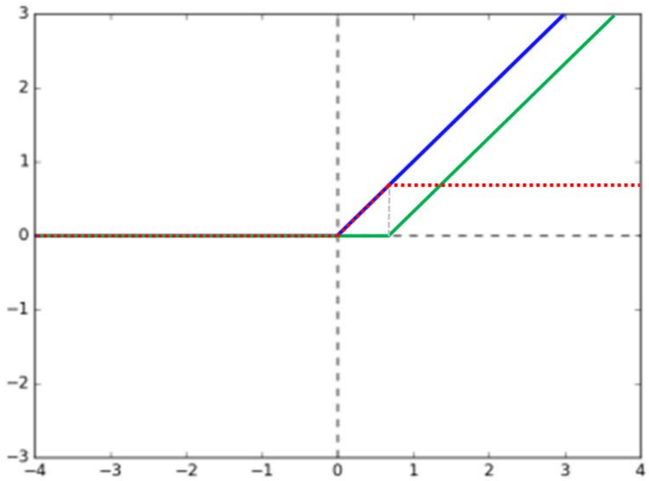
$$\frac{\partial\sigma}{\partial z}(0) = \sigma(0)\big(1 - \sigma(0)\big) = \frac{1}{2}\cdot\left(1 - \frac{1}{2}\right) = \frac{1}{4} < 1$$

כלומר, ככל שהרשת עמוקה יותר, כך נכפול את הגרדיאנט בעוד ערכים שקטנים מ-1 והשגיאה תדעך יותר. לבעיה הנ"ל ישנן מספר פתרונות. דרך אחת שפותרת את הבעיה היא להשתמש בשכבה לא ליניארית אחרת. השכבה שבה נשתמש, שהיא הנפוצה ביותר בימינו, נקראת Rectified Linear Unit או בקיצור ReLU.

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



לשכבה זו צורה מאוד שונה מהשכבות הקודמות. אף על פי כן, הרכבה של שתי פונקציות כאלו יכולות בקלות לקרב פונקציות בסגנון של סיגמואיד.

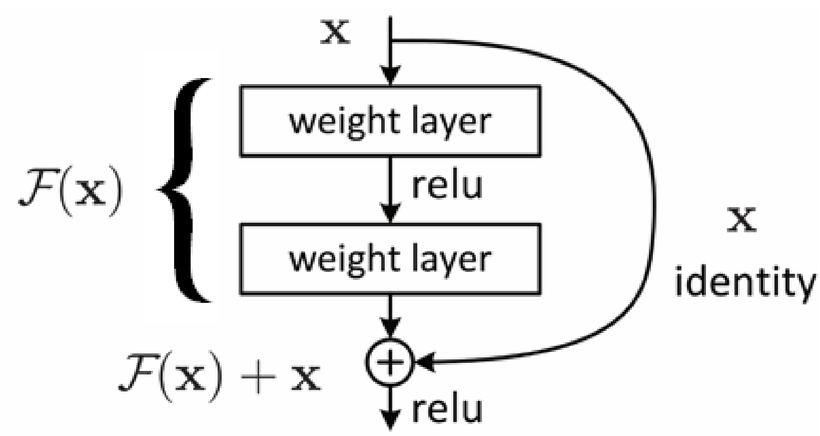


בכחול ובירוק ניתן לראות שני ReLU שונים, כאשר ה-ReLU הירוק מוזז. הזזה כזו יכולה להתקבל כאשר יש איבר הטיה בשכבה הקודמת (תזכורת: אם מדובר בשכבה לינארית המיוצגת ע"י $y = Ax + b$ אזי b הוא גורם ההטיה). באדום מצויר ההפרש בין הקו הכחול לקו הירוק. הפרש כזה יכול להיווצר כמכפלה וסכום של שכבה עוקבת בה המשקולות בערכים ± 1.

באזור בו הגרדיאנט אי-שלילי הערך שלו הוא קבוע בגודל 1. כלומר, ככל שנשרשר יותר שכבות גודל הגרדיאנט לא ישתנה.

שכבות שארית (Residual)

שכבת שארית (Residual layer) היא שכבה נפוצה שעושה פעולה מאוד פשוטה.



שכבות שארית מעבירות את אות הכניסה הלאה, בעזרת "חיבור מדלג" (skip connection), בתוספת לפעולה אחרת (פעולה לינארית או פעולת קונבולוציה).

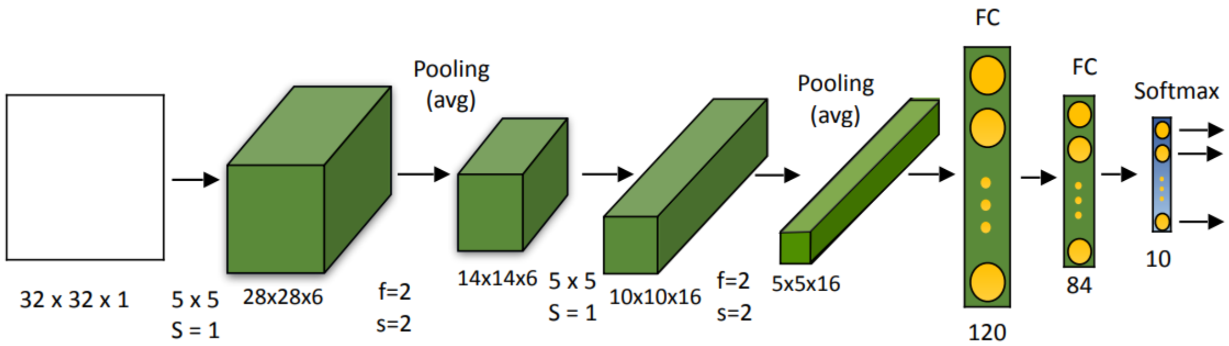
מוטיבציה אחת לשימוש בשכבות מסוג זה היא שהגרדיאנט של השכבה הוא:

$$\frac{\partial residual(x)}{\partial x} = \frac{\partial \mathcal{F}}{\partial x} + 1$$

ע"י שימוש בשכבה מסוג זה הגרדיאנט לא דועך במעבר בין שכבות הרשת.

רשתות CNN סטנדרטיות

רשת קונבולוציה שהיוותה בסיס לשכבות קונבולוציה עתידיות היא LeNet-5 (Y. LeCun, november 1998):



רשת זו אומנה לזהות ולסווג ספרות כתב יד על MNIST, המשימה שאותה ראיתם בניסוי הראשון.

הרשת מורכבת מהשכבות הבאות:

1. שכבת קונבולוציה עם 6 גרעיני קונבולוציה בגודל 5x5 (סה"כ 156 פרמטרים, כולל ה-bias של כל גרעין).
2. שכבת הורדת ממדיות average pooling עם פילטרים בגודל 2x2 וגודל צעד 2.
3. שכבת לא-לינארית מסוג tanh.
4. שכבת קונבולוציה עם 16 גרעיני קונבולוציה בגודל 5x5 (סה"כ 2416 פרמטרים).
5. שכבת הורדת ממדיות average pooling עם פילטרים בגודל 2x2 וגודל צעד 2.
6. שכבת לא-לינארית מסוג tanh.
7. שכבת קונבולוציה עם 120 גרעיני קונבולוציה בגודל 5x5 (סה"כ 48120 פרמטרים).
   שימו לב שהגודל המרחבי של תמונות הכניסה לשכבה זו הוא 5x5 כלומר גודל הפילטר הוא גודל התמונה! על כן במקרה זה פעולת הקונבולוציה למעשה שקולה לחלוטין לפעולה של שכבה לינארית (כפל במטריצה).
8. שכבה לינארית בגודל 84 (מספר פרמטרים 10164)
9. שכבה לינארית בגודל 10 (מספר פרמטרים 850).
10. שכבת softmax המעבירה את הערכים להסתברויות.
11. שכבת שגיאה מסוג Negative Log Likelihood.

שיטות אופטימיזציה מתקדמות

נזכיר את אלגוריתם האימון בו השתמשנו עד כה: gradient descent.

בהינתן פונקציה $f(x)$, בכל צעד, נחשב את הגרדיאנט של הפונקציה ולאחר מכן ננוע בכיוון הגרדיאנט כפול גודל הצעד.

$$g_k = \nabla f(x_k)$$

$$x_{k+1} = x_k + \eta_k g_k$$

כעת נציג מספר וריאציות על שיטה פשוטה זו שנפוצות באימון רשתות.

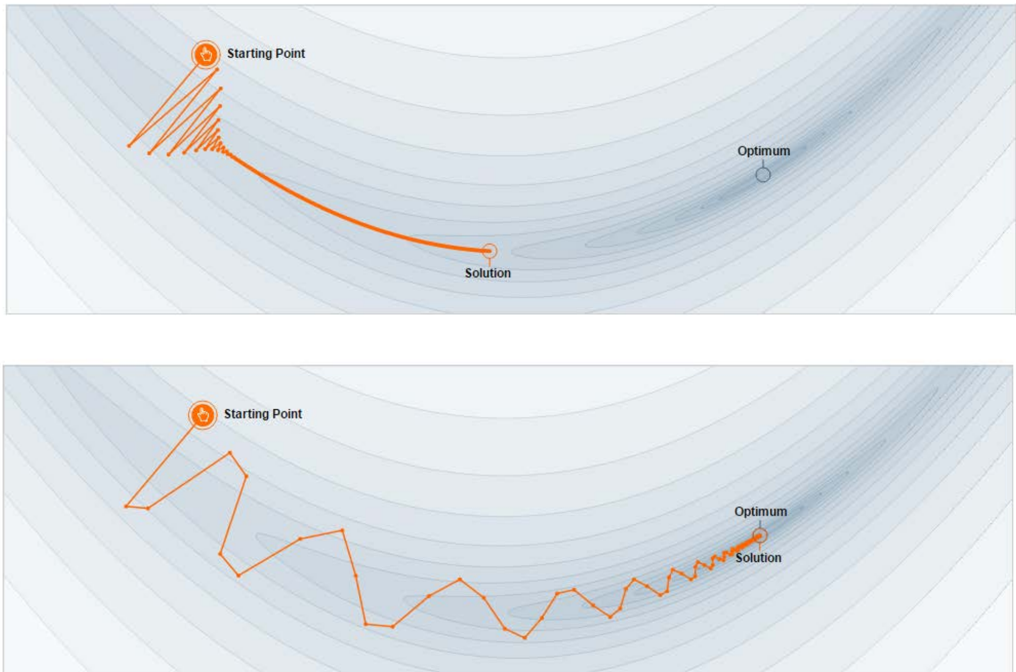
1. מומנטום

שיטת מומנטום, או בשמה המלא stochastic gradient descent with momentum, משנה את צעד העדכון כדי להתחשב בקצב שינוי הפרמטרים במרחב הפרמטרים.

$$v_{k+1} = \alpha v_k - \eta \nabla f(x_k)$$

$$x_{k+1} = x_k + v_{k+1}$$

האיבר α הוא קבוע קרוב ל-1 (בד"כ 0.9) המשמש לקבוע את מידת ההתחשבות בגרדיאנטים של העבר.
שינוי זה שקול לדחיפת כדור במורד מדרון (הגרדיאנט), כאשר הוא עם הזמן צובר תאוצה (מומנטום) אבל פועל עליו כח חיכוך עם האוויר (גודל הצעד).
ויזואלית ההבדלים נראים כך (למעלה ללא מומנטום, למטה עם מומנטום של 0.9):



אינטואיציה פיזיקלית

לאלו מכם שמרגישים בנח עם משוואות תנועה ניוטוניות, מצורף ההסבר מטה.

נתבונן בגרסה רציפה של gradient descent :

$$\frac{dx}{dt} = -\eta \nabla_x E(x)$$

כאשר $E(x)$ הוא פוטנציאל חיצוני (שנגזרתו היא הכוח הפועל על מסה) ו- $\frac{dx}{dt}$ מבטא את המהירות (קצב שינוי הפרמטרים).

הגרסה הרציפה של מומנטום מתאימה למשוואה הניוטונית המתארת תנועת מסה m בתווך צמיגי עם מקדם חיכוך μ תחת השפעת כוח בעל פוטנציאל $E(x)$:

$$m\frac{d^2x}{dt^2} + \mu\frac{dx}{dt} = -\nabla_X E(X)$$

ע״י דיסקרטיזציה של המשוואה נקבל :

$$m\frac{x_{t+\Delta t} + x_{t-\Delta t} - 2x_t}{(\Delta t)^2} + \mu\frac{x_{t+\Delta t} - x_t}{\Delta t} = -\nabla_x E(x)$$

$$x_{t+\Delta t} - x_t = -\frac{(\Delta t)^2}{m + \mu\Delta t}\nabla_x E(x) + \frac{m}{m + \mu\Delta t}(x_t - x_{t-\Delta t})$$

הביטוי הנ״ל זהה לביטוי המקורי של המומנטום.

2. Adaptive Gradient (AdaGrad)

AdaGrad היא שיטת gradient descent בה לכל פרמטר קצב לימוד משלו הנקבע על סמך היסטוריית הגרדיאנטים שלו. אלגוריתם זה נותן קצב לימוד גבוה לפרמטרים שמשתנים לעיתים רחוקות וקצב לימוד נמוך לפרמטרים שמשתנים לעיתים קרובות.

משוואת האלגוריתם נראית כך :

$$r_{k+1} = r_k + g \odot g$$

$$x_{k+1} = x_k - \frac{\eta}{\epsilon + \sqrt{r_{k+1}}} \odot g$$

כאשר $\odot$ מייצג מכפלה איבר-איבר, ϵ הינו גודל קטן שנועד למנוע בעיות נומריות.

בעיה אפשרית של שימוש ב-Adagrad היא העובדה שהווקטור r הולך וגדל לאורך האימון ולא מאפשר ״לשכוח״ גרדיאנטים שראה בעבר הרחוק.

3. RMSprop

שיטת RMSprop מתמודדת עם הבעיה הנ״ל ב-Adagrad ע״י החלפת העדכון של r בממוצע נע של r כך :

$$r_{k+1} = \rho r_k + (1-\rho) g \odot g$$

$$x_{k+1} = x_k - \frac{\eta}{\epsilon + \sqrt{r_{k+1}}} \odot g$$

כאשר $\rho = 0.9$ בד״כ.

4. Adaptive Moment Estimation (Adam)

Adam הוא אלגוריתם שמנסה לשלב בין מומנטום לבין שיטות גרדיאנט אדאפטיביות:

$$m_{t+1} = \beta_1 m_t + (1-\beta_1)g$$

$$v_{t+1} = \beta_2 v_t + (1-\beta_2)g\odot g$$

$$x_{k+1} = x_k - \frac{\eta}{\epsilon + \sqrt{v_{t+1}}}\odot m_{t+1}$$

m_t ו- v_t הם שערוכים של המומנטים מסדר ראשון ושני (תוחלת ושונות) של הגרדיאנטים בהתאמה, ומכאן נובע השם.

אבל לשיטה הנ"ל יש בעיה: מכיוון שמאתחלים את הווקטורים m_t ו- v_t לווקטורי אפסים, בהמשך תהליך האימון ל- m_t ול- v_t יש נטייה להישאר קרובים ל-0.

כדי להתמודד עם הבעיה זו מבצעים את התיקון הבא:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$x_{k+1} = x_k - \frac{\eta}{\epsilon + \sqrt{\hat{v}_{t+1}}}\odot \hat{m}_{t+1}$$

מעשית ברוב הרשתות המודרניות משתמשים ב-Adam או ב-SGD + Momentum.

רגולריזציה

ברשתות מודרניות יש מליוני פרמטרים, לעיתים יותר ממספר הדוגמאות שרואים באימון!

כתוצאה מכך, רשתות נוטות להגיע ל-overfitting, כלומר להתאים את עצמן בצורה כמעט מושלמת לדוגמאות שהן ראו על חשבון ההכללה לדוגמאות שלא נראו בסט האימון.

רגולריזציה מוגדרת להיות כל שינוי שעושים ברשת שנועד להקטין את שגיאת ההכללה אך לא את שגיאת האימון.

שיטות רגולריזציה בד"כ מעודדות מודלים פשוטים יותר בהתאם לעיקרון Occam's razor.

דוגמה לרגולריזציה שראינו היא שימוש ברשתות קונבולוציה במקום רשתות לינאריות fully connected שמשתמשות בידע שיש לנו על מבנה של תמונות טבעיות ועל האופן שבו ניתן לחלץ מאפיינים שלהן.

כעת נציג מספר כלים מארגז הכלים שיש לנו כדי להתמודד עם התופעה.

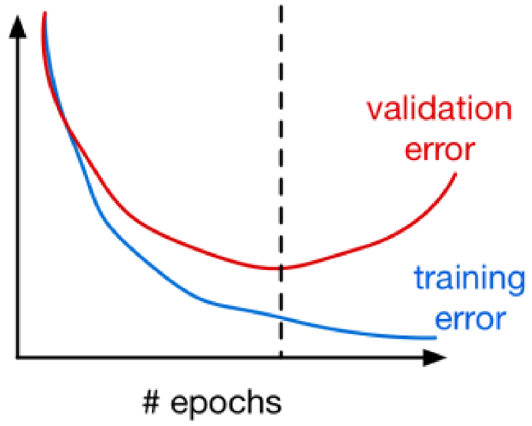
1. Early Stopping

אחת הדרכים הפשוטות להימנע מ-overfitting היא לעצור את תהליך האימון ברגע שאנחנו לא רואים יותר שיפור ביכולת ההכללה של הרשת.

ניתן לזהות נקודה זו ע"י בחינת השגיאה של הרשת על סט הוולידציה ועצירת האימון כאשר שגיאה זו גדלה.

ניתן גם לבצע "עצירה בדיעבד". כלומר, להמשיך לאמן, ואז להשתמש במודל כפי שהיה מצבו בנקודה שזיהינו כטובה לעצירה.

שימו לב- כלי רשתות הנוירונים של MATLAB מבצע Early stopping באופן דיפולטי. השיטה בה הם פועלים היא בדיקה של המגמה של סט הולידציה- במידה וה-validation error גדל במשך מספר מסוים של בדיקות, התהליך נעצר.



איור 17 – הנקודה באימון בה נחליט לעצור כדי להימנע מ-overfitting

2. <u>פחות פרמטרים</u>

השיטה הטריוויאלית והמתבקשת היא הקטנת מספר הפרמטרים של הרשת ושימוש ברשת קטנה יותר. למשל ע"י מעבר לשכבות קונבולוציה, הקטנת מספר המסננים, פחות שכבות ברשת.

3. <u>Weight Decay</u>

רגולריזציה מסוג weight decay, שגם מכונה ridge או L2, מגבילה את מרחב האפשרויות של משקולות הרשת ע"י "הענשה" על שימוש במשקולות גדולים מידי.

עבור פונקציית השגיאה הריבועית (כמו ב-Adaline), פונקציית המחיר החדשה תיראה בצורה הבאה:

$$Err(f) = \frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2 + \frac{\alpha}{2}\|w\|_2^2$$

עדכון המשקולות, עפ"י הגרדיאנט החדש ייראה כך:

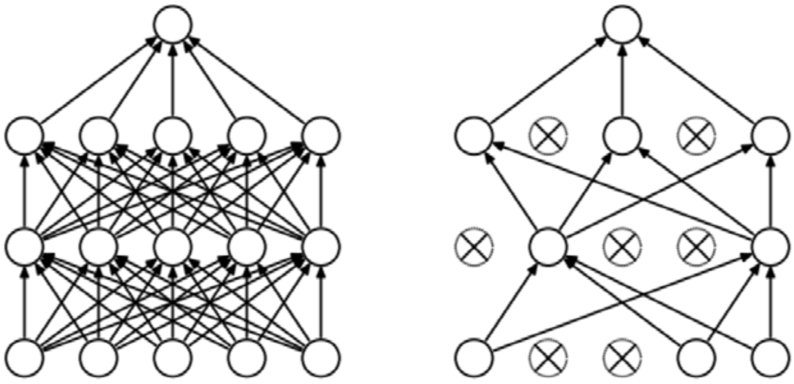
$$w^{t+1} = w^t - \eta_t\left(x\left({w^t}^T x - y\right) - \alpha w\right) = (1 - \eta_t\alpha)w - \eta_t\left({w^t}^T x - y\right)$$

קיבלנו כי עדכון המשקולות זהה לעדכון הקודם למעט דעיכה של המשקולות בפקטור $(1 - \eta_t\alpha)$, ומכאן מגיע השם weight decay.

4. <u>Dropout</u>

Dropout היא שיטת רגולריזציה נוספת לרשתות בה "מכבים" נוירונים בכל איטרציית אימון אקראית. עושים זאת ע"י הגדרת מטריצת מסיכה בינארית בה כל איבר מתפלג ברנולי (הטלת מטבע) עם פרמטר שנקבע ע"י המשתמש.

כלומר בכל איטרציה נוירון אחר "נכבה". האינטואיציה לשימוש בשיטה זו היא שכעת על הרשת ללמוד לסווג באמצעות מסלולים שונים ברשת והיא לא יכולה להסתמך על מספר קטן של נוירונים. נדגיש שאת שיטת dropout מפעילים במהלך האימון בלבד.



איור 18 – רשת לינארית עם ובלי dropout

5. Data Augmentation

הדרך הטובה ביותר לשפר את יכולת ההכללה של המודל היא לאסוף עוד דוגמאות! אך לרוב זה לא אפשרי ואנחנו מוגבלים בכמות הדוגמאות שיש לרשותנו (או שהתהליך מאוד יקר).

Data augmentation מאפשר להגדיל את אוסף הדוגמאות שלרשותנו ע"י ביצוע שינויים בדוגמאות הקיימות כך שיהוו דוגמאות אחרות.

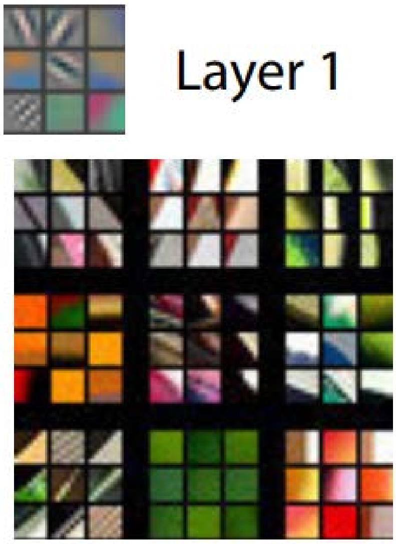
דוגמאות לשינויים נפוצים לתמונות כוללות: טרנסלציה (הזזה), שיקוף, סיבוב והוספת רעש.

מה נלמד בשכבות הפנימיות ברשת

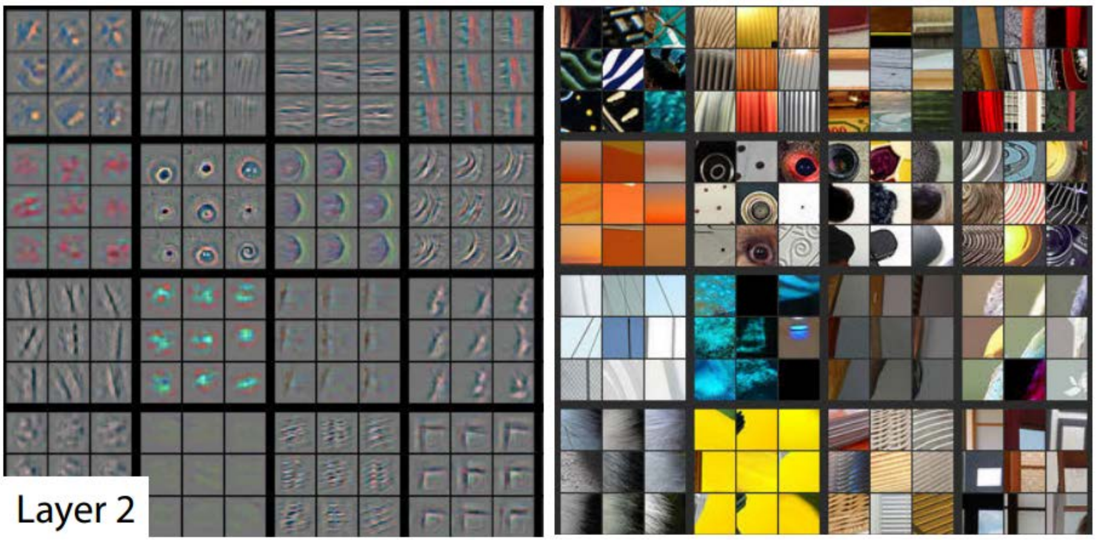
מתוך ניסיון להבין כיצד רשתות נוירונים עמוקות מצליחות לפתור בעיות מורכבות (בעיקר בעיות תמונה), החלו לחקור מה נלמד בכל שכבה ברשת .

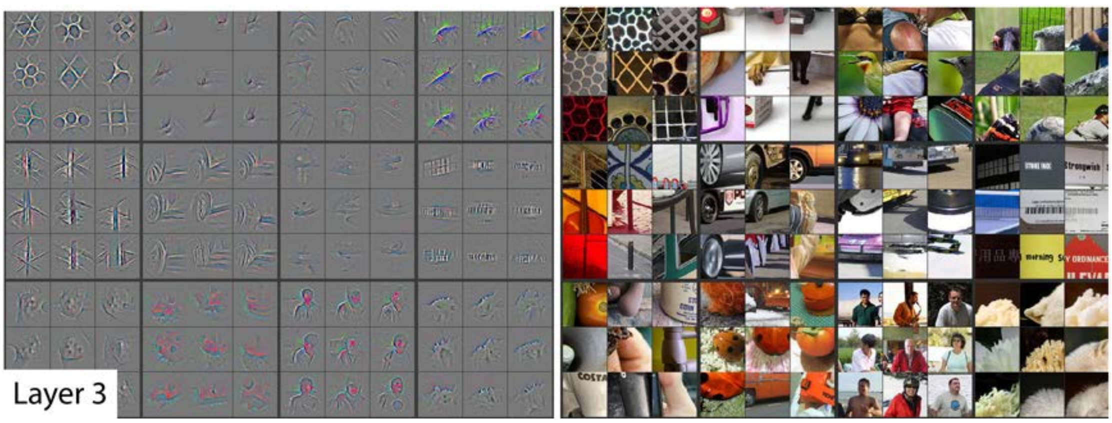
בתהליך זה גילו כי שכבות נמוכות לומדות לזהות מאפיינים בסיסים של תמונות כמו שפות ופינות, שכבות אמצעיות מזהות טקסטורות וצורות גאומטריות ושכבות גבוהות יותר מזהות עצמים מורכבים יותר כמו עיניים, פרצופים של בני אדם ושל בע"ח.

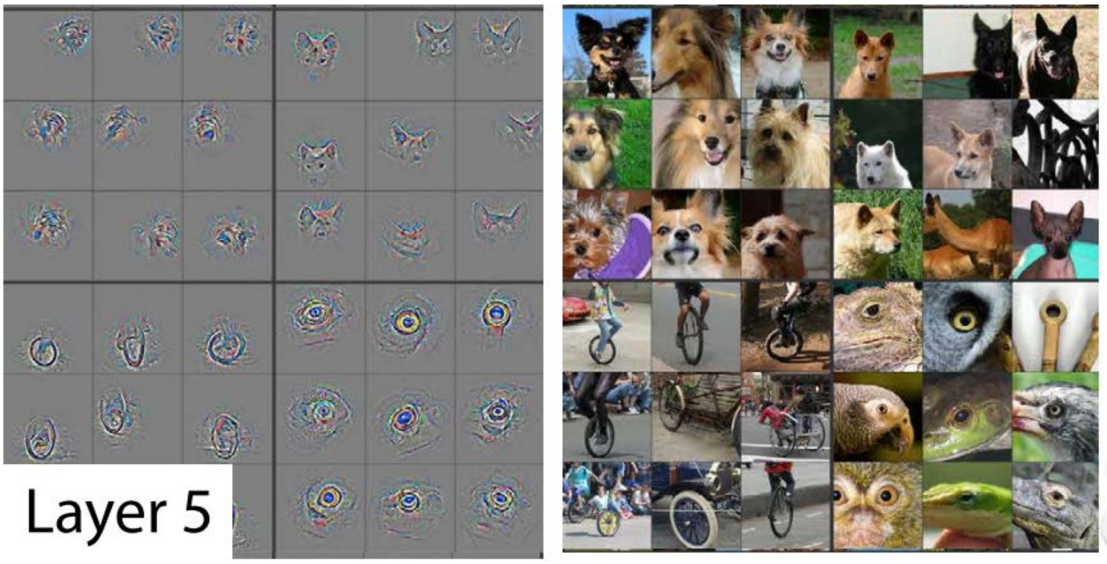
ויזואליזציה של שכבות בעומקים שונים ברשת שאומנה על סיווג תמונות ניתן לראות באיור 19. מימין רואים את תמונות המקור שהוזנו לרשת ומשמאל את האקטיבציות (פלט של שכבה ברשת לאחר השכבה הלא לינארית) של שכבות בעומקים שונים ברשת.



איור 19 – אקטיבציות של שכבה ראשונה ברשת סיווג







המעוניינים ללמוד עוד על הנושא (או סתם לצפות בתמונות מרהיבות בצורה אינטראקטיבית) מוזמנים להיכנס לכאן.

Transfer learning

Transfer learning מוגדר כך :

Transfer learning and domain adaptation refer to the situation where what has been learned in one setting … is exploited to improve generalization in another setting
— Page 526, Deep Learning, 2016.

כפי שראינו בסעיף הקודם, בשכבות שונות של הרשת נלמדים מאפיינים שונים של הקלט שלנו. אם ניקח את הרשת ונקטע אותה בשכבת ביניים, למעשה קיבלנו פונקציה שמטילה תמונה למרחב אחר בה יש **משמעות סמנטית** שניתן לנצל לטובת סיווג.
אך ניתן להשתמש באותה הפונקציה גם כדי להטיל את מרחב הדוגמאות למרחב אחר, בו נוכל לפתור בעיות סיווג ורגרסיה שונות מאשר המשימה המקורית עליה אימנו את הפונקציה.
כיצד עושים זאת? את השכבות העמוקות של הרשת, שלמדו לסווג דוגמאות למשימה ספציפית, נחליף בשכבות חדשות – אותן נאמן על דוגמאות חדשות התואמות את המשימה החדשה. השכבות הראשונות של הרשת ישארו ללא שינוי, ולא יעברו אימון נוסף ע"י הדוגמאות החדשות (ישארו "קפואות").
בחלק השני של הניסוי נדגים כיצד ניתן לקחת רשת שהתאמנה לבעיית סיווג של תמונות טבעיות ולפתור באמצעותה בעיה של סיווג של תמונות של קינוחים.

מפגש שני

שאלות הכנה

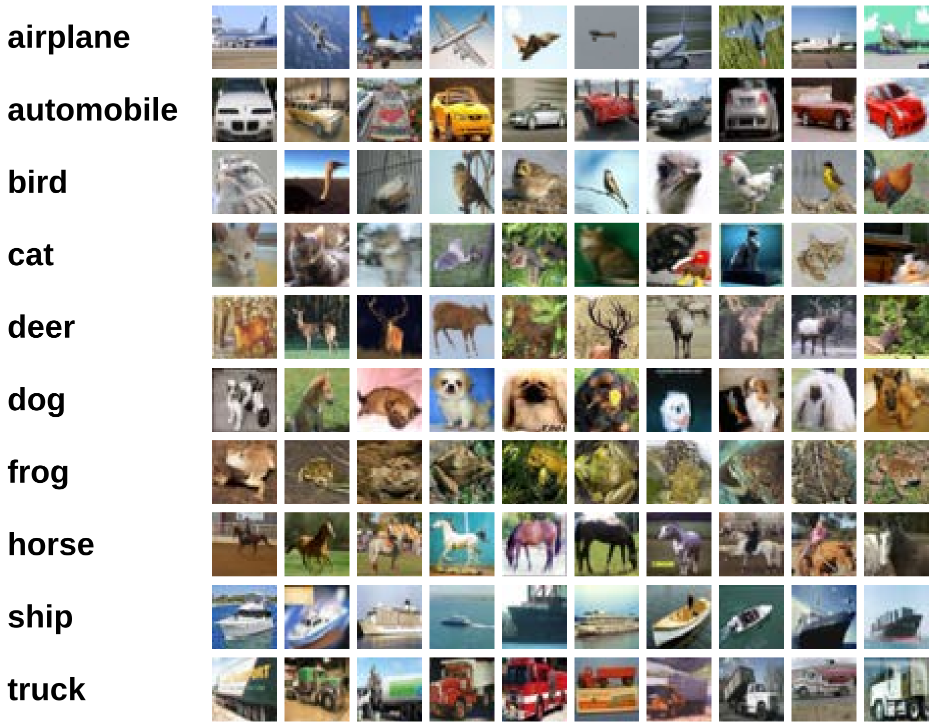
1. נתונה תמונת כניסה לרשת עם מימדים h_{in}, w_{in}, c_{in}.
שכבת הקונבולוציה הראשונה בעלת הנתונים הבאים:
   - מסננים בגודל k_h, k_w
   - מספר המסננים הוא d
   - אין ריפוד או דילוג.

   כתבו ביטויים ל- $h_{out}, w_{out}, c_{out}$, מימדי תמונת המוצא משכבת הקונבולוציה כפונקציה של הפרמטרים לעיל.
2. שיטות אופטימיזציה מתקדמות
בהינתן הפונקציה $f(x) = x^2$ ונקודת התחלה $x_0 = -1$, הדגם שלוש איטרציות של gradient descent עם מומנטום.
א. עבור גודל צעד $\eta = 0.01$.
ב. עבור גודל צעד $\eta = 2$.
3. Weight Decay
באיזה אופן משפיעה רגולריזציית L2 על המשקולות? התייחסו לפרמטר α בערכי קיצון ($\alpha \to 0$, $\alpha \to \infty$).
4. מהם היתרונות של CNN על פני fully connected במשימות סיווג תמונות?

מפגש שני – מהלך הניסוי

חלק א' – סיווג תמונות CIFAR10

בחלק זה נתמודד עם משימת סיווג תמונות ממאגר CIFAR10. מאגר זה מכיל תמונות בקטגוריות שונות, כאשר כל אחת מתויגת על ידי הקטגוריה המתאימה. הקטגוריות הן :



איור 20 - הקטגוריות במאגר cifar10 ודוגמאות לתמונות מכל קטגוריה
לקוח מתוך https://www.cs.toronto.edu/~kriz/cifar.html

נרצה לבנות רשת נוירונים המסווגת תמונות לקטגוריות המתאימות ב-CIFAR10. לשם כך נבנה ארכיטקטורה המשלבת שכבות קונבולוציה ושכבות ליניאריות.

- ❖ הריצו את החלק הראשון בקוד cifar10ClassificationDL אשר מוריד את CIFAR10. עדכנו את הנתיב לתיקייה המתאימה.

בשלב התאמת הפרמטרים של הרשת, מטעמי חיסכון בזמן, נסווג רק 4 מחלקות מתוך 10 המחלקות של CIFAR10. בסוף התאמת הפרמטרים נבצע אימון סופי על 10 המחלקות.

- ❖ עברו על הקוד והשלימו את השכבות הרלוונטיות לפי ההוראות. השכבות הנחוצות לחלק זה נמצאות בנספח א' – Matlab Layers חלק א', נספח ב' – Matlab Layers חלק ב'.

הערה : ישנן אופטימיזציות שונות הגורמות לירידה בגודל הצעד ככל שמתקדמים בתהליך הלמידה, ובכך מביאות לדיוק גבוה יותר בהתכנסות. בחלק זה של המעבדה נשתמש בכזאת אופטימיזציה, אשר מקטינה

את גודל הצעד בפקטור קבוע כל כמות epoch-ים שנקבעת מראש. ניתן לראות את השימוש בחלק הבא באופציות האימון : 'LearnRateSchedule', 'LearnRateDropFactor', 'LearnRateDropPeriod'.

נבחן את השפעות הוספת רגולריזציה על הרשת.

כפי שאתם יכולים לראות בקוד, כבר קיימת רגולריזציית L2.

❖ הריצו את הרשת עם הפרמטרים הקיימים וצרפו את תוצאות הריצה.
❖ הגדילו את ערך רגולריזציית L2 ל-0.5. צרפו את תוצאות הריצה.

1. מה קורה לתהליך ההתכנסות? מה הסיבה? בחרו את הערך המתאים ביותר והמשיכו איתו הלאה.

❖ הוסיפו לרשת שכבת dropout יחידה, בסוף הקונבולוציות מיד לפני שכבות ה-fullyConnected.

2. הסבירו מהי משמעות ערך ה-probability ב-dropout.

❖ הריצו את הרשת עם לפחות 3 ערכים שונים של dropout probability בטווח 0-0.5, כל פעם במשך 20 אפוקים.

3. מה ההשפעה של dropout על ההתכנסות? הסבירו והוסיפו נתונים וגרפים מתאימים. בחרו את הערך המתאים ביותר והמשיכו איתו הלאה.

כעת, נוסיף Data augmentation.

4. איזה סוג Data augmentation מתאים כאן? הסבירו בקצרה בעזרת דוגמאות.

❖ הסתכלו בתחילת הקוד על השורות אשר בהערות המתאימות לסעיף ה-Data augmentation. הוסיפו אותן לקוד, והגדירו את ה-Data augmentation אשר נראה בעיניכם מתאים. היעזרו בתיעוד של MATLAB עבור האופציות הקיימות ב-imageDataAugmenter. שימו לב שעליכם להגדיר את הaugmenter כפרמטר בaugmentedImageDatastore. הריצו אימון מחדש וצרפו את התוצאות לדוח.

5. איך ה-Data augmentation השפיע על התוצאות? הישארו עם השינויים שתרמו.

לסיום, נבחן את השפעת גודל הצעד.

6. שנו את גודל הצעד בכמה דרכים (למשל הכפילו ב-2, חלקו ב-2 וכו'). מצאו את גודל הצעד המיטבי והישארו איתו.

בטלו את ההגבלה ל-4 מחלקות מתוך ה-10 והריצו אימון מחדש עבור סיווג לכל המחלקות.

צרפו לדו"ח שלכם את התוצאות שמתקבלות.

חלק ב' – Transfer Learning

בחלק זה נאמן רשת נוירונים לסיווג סוגי קינוחים.

ה-Dataset למשימה זו מורכב מתמונות, ותיוג של כל תמונה אשר מעיד על המאכל בתמונה.

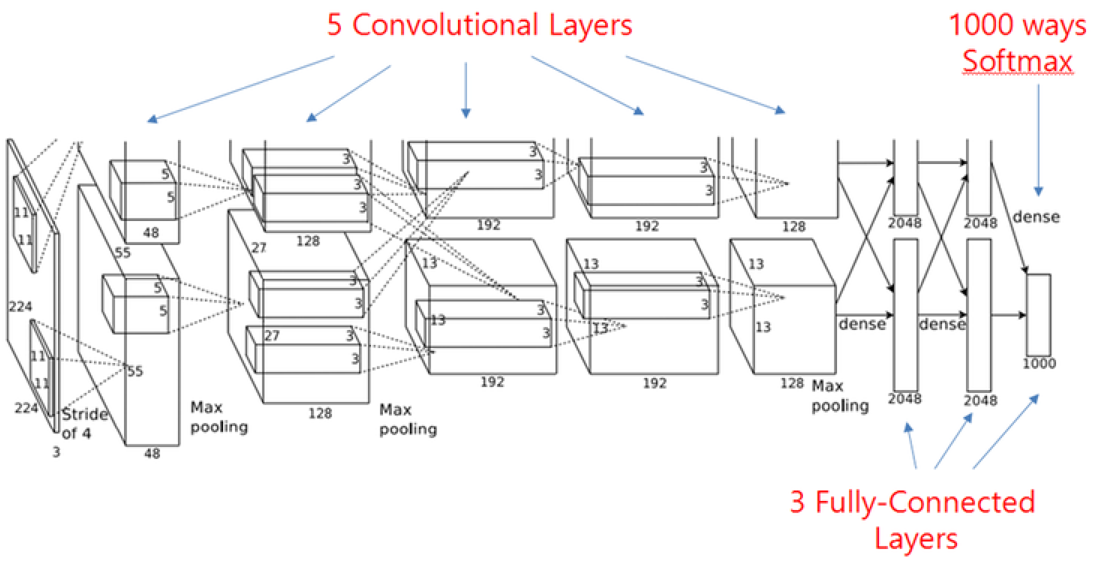
כמות המידע המתויג אשר ניתנת לכם בתרגיל זה היא קטנה, ונרצה לנצל כל ידע מוקדם שקיים לנו על מנת לשפר את ביצועי המערכת. לשם כך, נשתמש ב-Transfer Learning – ניקח רשת שאומנה מראש למשימת סיווג תמונות (בדומה לרשת מהסעיף הקודם), ונבנה על בסיס הרשת המאומנת מודל חדש אשר יורחב לפתרון הבעיה הייעודית החדשה. לבסוף, נאמן את המודל החדש בעזרת ה-Dataset החדש ונבחן את תוצאות הרשת החדשה שבנינו ואימנו.

ניתן למצוא באינטרנט רשתות שונות שאומנו לפתור בעיות שונות, אשר ניתנות להורדה. למשל, עבור בעיית הסיווג עמה התמודדנו בסעיף הקודם ניתן למצוא רשתות בארכיטקטורות שונות (AlexNet, GoogLeNet ועוד) המאומנות מראש להורדה זמינה.

בדוגמה זו נשתמש ברשת AlexNet לסיווג תמונות, אשר אומנה על מאגר חלקי מתוך ImageNet (מידע נוסף על מאגר ImageNet ניתן למצוא [כאן](#)).

❖ גירסה מאומנת של AlexNet עבור MATLAB ניתנת להורדה [כאן](#).

Alexnet היא הרשת אשר הובילה למהפכה ב-2012 כאשר הצליחה לסווג את מאגר ImageNet בדיוק של מעל 10% מכל שאר המתחרים במשימה. הרשת מורכבת מ-5 שכבות קונבולוציה ו-3 שכבות Fully-Connected. על מבנה הרשת והחידושים שבה תוכלו לקרוא [פה](#).



איור 20 - מבנה הרשת Alexnet

על מנת להתאים את הרשת למשימת הסיווג שלנו, ניקח את הרשת המאומנת מראש ו"נקפיא" את השכבות שלה פרט ל-3 שכבות האחרונות, אותן נחליף לשכבות חדשות ונאמן על המאגר החדש עבור המשימה החדשה. לבסוף, הרשת שנקבל תהיה מורכבת מ-22 שכבות עם משקולות קבועות (בצבע כחול), ו3 שכבות אשר המשקולות שלהן יילמדו בתהליך אימון מחדש (בצבע אדום).

```
' 1   data'    Image Input
' 2   conv1'   Convolution
' 3   relu1'   ReLU
' 4   norm1'   Cross Channel Normalization
' 5   pool1'   Max Pooling
' 6   conv2'   Convolution
' 7   relu2'   ReLU
' 8   norm2'   Cross Channel Normalization
' 9   pool2'   Max Pooling
' 10  conv3'   Convolution
' 11  relu3'   ReLU
' 12  conv4'   Convolution
' 13  relu4'   ReLU
' 14  conv5'   Convolution
' 15  relu5'   ReLU
' 16  pool5'   Max Pooling
' 17  fc6'     Fully Connected
' 18  relu6'   ReLU
' 19  drop6'   Dropout
' 20  fc7'     Fully Connected
' 21  relu7'   ReLU
' 22  drop7'   Dropout
' 23  fc8'     Fully Connected
' 24  prob'    Softmax
  25  'output' Classification Output
```

בתיקייה הרלוונטית לחלק זה מצורפים לכם שני קטעי קוד חלקיים-

- TransferLearning.m – הפעלת המודול לאימון ובדיקת הרשת למשימה החדשה.
- freezeWeights.m – פונקציה המקבלת אוסף שכבות ומחזירה גרסה "מוקפאת" של השכבות.

א. מהם היתרונות של Transfer Learning? באילו סוגים של משימות ניתן להשתמש בשיטה זו?

ב. מהו אופי הפלט של הריצה שאתם מצפים לקבל עם שימוש ב-Transfer Learning לעומת ריצה המאמנת רשת שלמה מנקודת מוצא נקיה?

❖ השלימו את שורות הקוד במקומות המתאימים בקובץ TransferLearning.m.
הוסיפו לדוח שלכם את פלט ההרצה והגרף המתקבל.

❖ השתמשו בידע שרכשתם במעבדה זו על מנת לנסות למקסם את תוצאות ה-Transfer Learning. שחקו עם הארכיטקטורות והפרמטרים באיזו צורה שתחפצו. הסבירו בדוח מה ניסיתם לעשות וצרפו תוצאות שקיבלתם.

נספח א' – Matlab Layers חלק א'

טבלה מסכמת של השכבות הנחוצות לשימוש במפגש א' (לקוח מהתיעוד באתר של MathWorks).

Input Layers

Function	Description
imageInputLayer	An image input layer inputs images to a network and applies data normalization.

Learnable Layers

Function	Description
fullyConnectedLayer	A fully connected layer multiplies the input by a weight matrix and then adds a bias vector.

Activation Layers

Function	Description
reluLayer	A ReLU layer performs a threshold operation to each element of the input, where any value less than zero is set to zero.

Output Layers

Function	Description
softmaxLayer	A softmax layer applies a softmax function to the input.
classificationLayer	A classification output layer holds the name of the loss function the software uses for training the network for multiclass classification, the size of the output, and the class labels.
regressionLayer	A regression output layer holds the name of the loss function the software uses for training the network for regression, and the response names.

נספח ב' – Matlab Layers חלק ב'

טבלה מסכמת של השכבות הנחוצות לשימוש במפגש ב' **בנוסף** לשכבות המוצגות בנספח א' (לקוח מהתיעוד באתר של MathWorks).

Function	Description
`convolution2dLayer`	A 2-D convolutional layer applies sliding filters to the input. The layer convolves the input by moving the filters along the input vertically and horizontally and computing the dot product of the weights and the input, and then adding a bias term.

Normalization and Dropout Layers

Function	Description
`dropoutLayer`	A dropout layer randomly sets input elements to zero with a given probability.

Pooling Layers

Function	Description
`averagePooling2dLayer`	An average pooling layer performs down-sampling by dividing the input into rectangular pooling regions and computing the average values of each region.
`maxPooling2dLayer`	A max pooling layer performs down-sampling by dividing the input into rectangular pooling regions, and computing the maximum of each region.
`maxUnpooling2dLayer`	A max unpooling layer unpools the output of a max pooling layer.