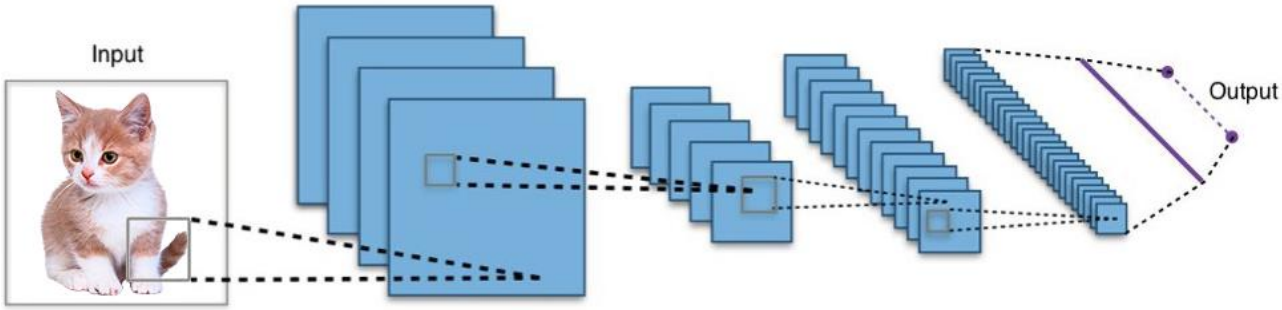
הטכניון – מכון טכנולוגי לישראל הפקולטה להנדסת חשמל ומחשבים ע״ש ויטרבי

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.
אורי בריט ורותם מוליוף – נובמבר 2020, פברואר 2021.
אורי בריט – אפריל, יולי 2021.

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, המתכוננים 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 האימון (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 האימון**) $\{x_i^t, y_i^t\}_{i=1}^m$. סט זה משמש להערכת הביצועים של המסווג הנלמד. בשלב הערכת הביצועים נפעיל את המסווג על וקטורי המאפיינים $\{x_i^t\}_{i=1}^m$ ונשווה את התיוגים של המסווג לתיוגים הנכונים $\{y_i^t\}_{i=1}^m$.

  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 האימון וסדרת הבוחן שלנו הן בעלות אותו פילוג ובלתי תלויות (i.i.d. = Independent and identically distributed random variables). כלומר :

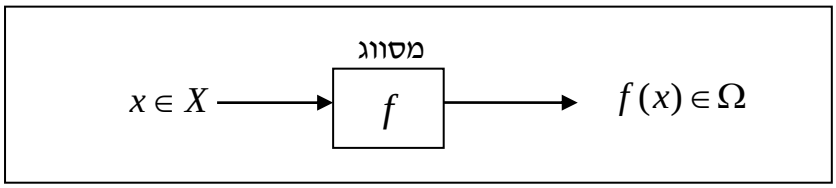
$$p(x_i) = p(x_j), \qquad \forall x_i, x_j \in \{training \backslash test\ set\}$$

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

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

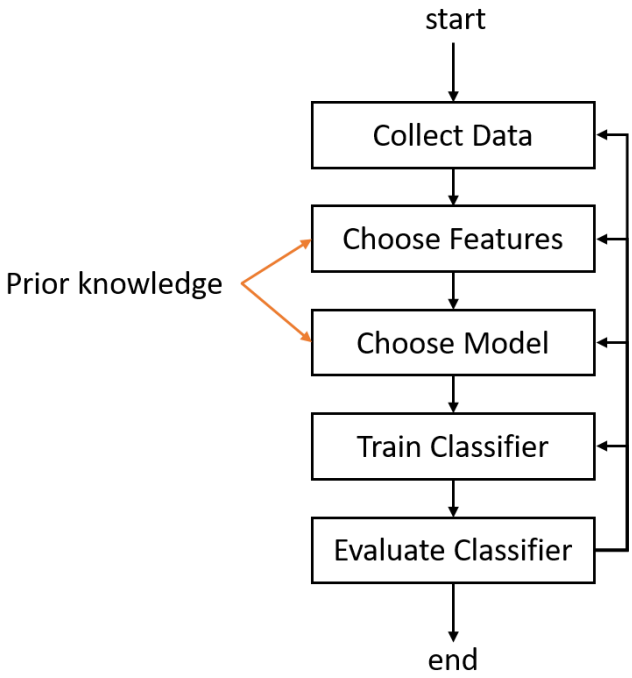
בהינתן סדרת אימון $\{x_i, y_i\}_{i=1}^{n}$, נרצה למצוא מסווג $f : X \to \Omega$ כך שיסווג את סדרת הבוחן $\{x_i^t\}_{i=1}^{m}$ למחלקה המתאימה עם שגיאה קטנה ככל הניתן.

באופן סכמתי מסווג מוגדר בצורה הבא:



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

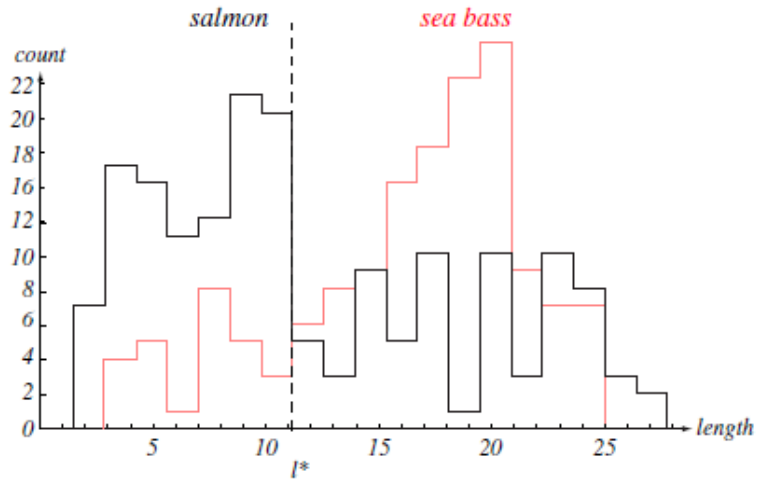
התרשים הבא מתאר בצורה סכמתית את תהליך האימון בבעיית הסיווג.



איור 1 – תיאור סכמתי של תהליך האימון

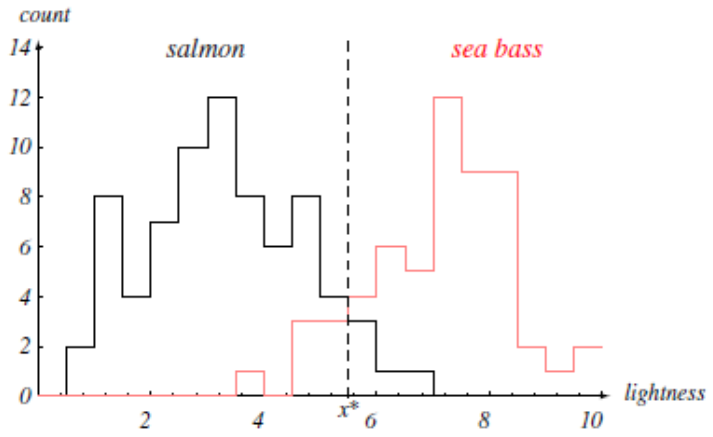
לאחר שלב איסוף המידע (תמונות של דגים עם תיוג), יש לבחור את המאפיינים הרלוונטיים לתיאורו ואת המודל המתאים למערכת (אורך ובהירות); בשלב זה ניתן לשלב ידע מקדים אודות המערכת. לאחר מכן מגיע שלב אימון המסווג ובחינת ביצועיו.
הידע המקדים אותו אנחנו מכניסים למערכת בדוגמת הדגים הינה שניתן להסתפק בבהירות ואורך הדג 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

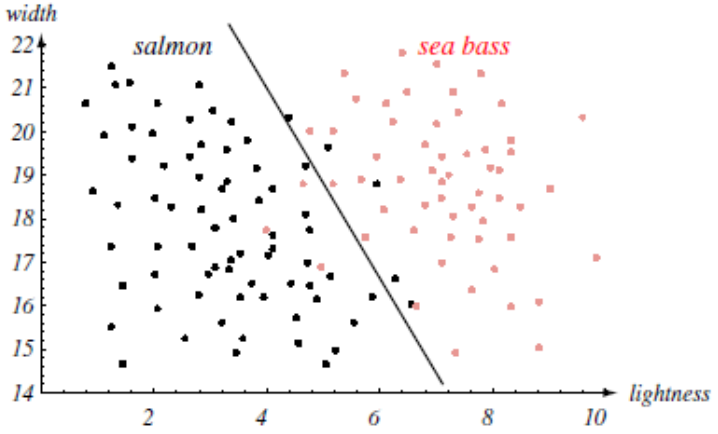
נשאלת השאלה האם שימוש בשני המאפיינים יחד יכול להביא לייצוג טוב יותר של הבעיה. על מנת לענות על שאלה זו, ניתן לצייר את ערך המאפיינים בתרשים דו-ממדי, כפי שמוצג באיור 4. כפי שניתן לראות,

[2] כל האיורים נלקחו מתוך:

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

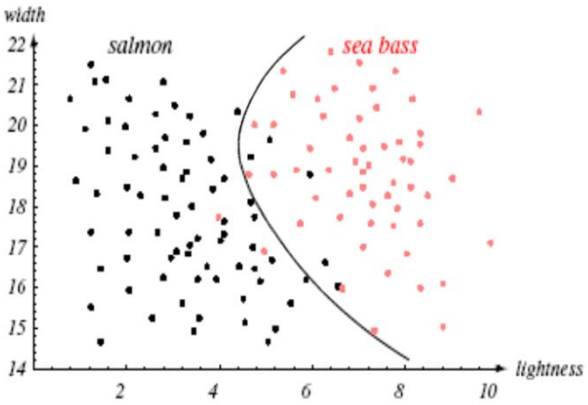
ונערכו לצורך התאמתם לתוכן החוברת.

ניתן להעביר קו ישר בין שתי המחלקות המסווג באופן נכון את רוב דוגמאות סדרת האימון. ניתן להבחין שבעזרת הייצוג הדו-ממדי שתי המחלקות ניתנות להפרדה טובה יותר מאשר בשימוש באחד מהמאפיינים.

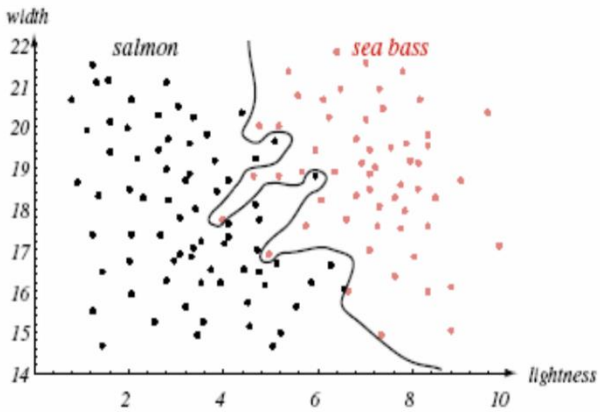


איור 4 – הצגה דו-ממדית של המאפיינים עם מסווג לינארי

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 5 – מסווג פולינומיאלי



איור 6 – מסווג הגורם להתאמת יתר

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) $\{x_i^t, y_i^t\}_{i=1}^m$:

$$Err(f) = \frac{1}{m}\sum_{i=1}^{m} \delta\left(f(x_i^t), y_i^t\right)$$

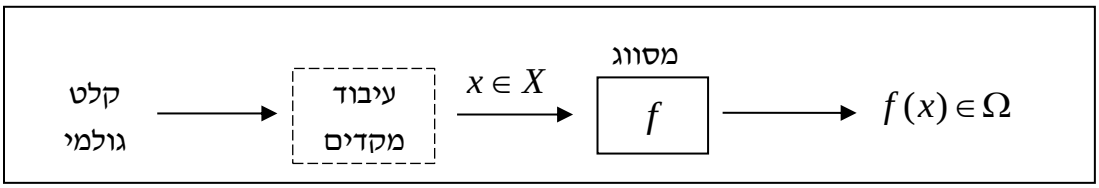
כאשר δ הינה פונקציית השגיאה (נקראת zero-one loss) : $\delta\left(f(x_i^t), y_i^t\right) = \begin{cases} 0 & f(x_i^t) = y_i^t \\ 1 & f(x_i^t) \neq y_i^t \end{cases}$.

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 האימון חיוני לטובת :

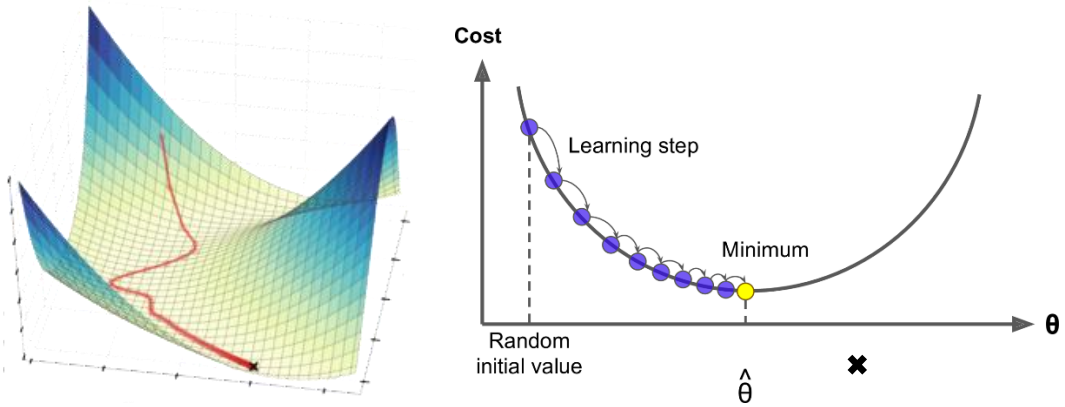
1. התאמת הקלט למודל הלמידה. כלומר, ישנם סוגי מסווגים המתאימים לקלט מסוג מסוים.
2. הורדת ממדיות הבעיה והפיכתה לפשוטה יותר.
3. האצת שלב הלמידה או האימון של המסווג.
4. שיפור רמת הביצועים של המסווג.

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:

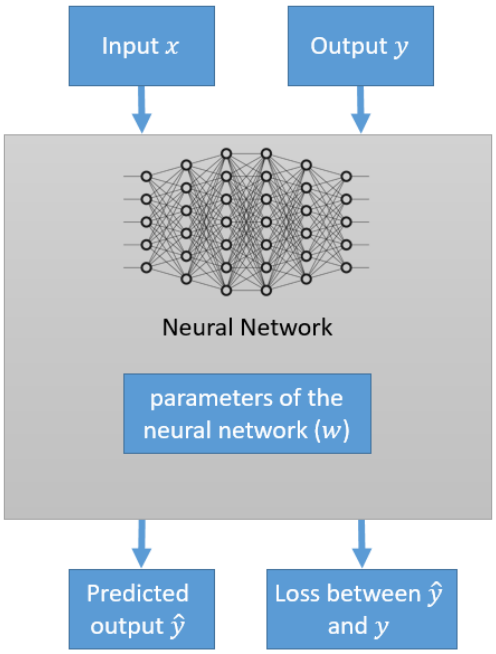
- x נקודת מידע כלשהי (תמונה, נתונים וכו')
- y תיוג מתאים לנקודה x

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 (פונקציית ה-Loss)**.

לאחר שהמערכת אומנה, נרצה לבצע הסקה על דוגמה חדשה. נזין לכניסת המערכת את הדוגמה הרלוונטית x_{new}, והמערכת תשתמש בפרמטרים אשר נלמדו בשלב הקודם על מנת לחזות $\hat{y}_{new}$ עבור x_{new}.

באופן זה ניתן לבצע חיזוי עבור דוגמאות שאנו לא יודעים את תיוגן, בעזרת למידה מדוגמאות מתויגות נתונות.

<u>נוירונים?</u>

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 האימון, שלב בחירת המודל נקבע ע״י בחירת **פונקציית השגיאה** ובחירת **מבנה (ארכיטקטורת)** הרשת. שלב אימון המודל נעשה ע״י Gradient Descent כדלקמן:

<u>אלגוריתם אימון של רשת נוירונים</u>

<u>קלט</u>: סדרת דוגמאות מתויגות לאימון $\{(x_i, y_i)\}_{i=1}^{n}$, פרמטר קצב האימון $\eta_0 > 0$,
מספר חזרות מרבי על סדרת האימון K.

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

<u>שלבי האימון</u>:

1. אתחול: אתחלו את הווקטור w לערך כלשהו.
2. עבור החזרה ה-k $(k = 1,2,3, \dots, K)$:
   2.1. קבעו את קצב האימון η_k עבור האיטרציה הנוכחית.
   2.2. עברו על דוגמאות סדרת האימון $i = 1,2,3, \dots, n$:
      2.2.1. קבלו את הווקטור x_i והתיוג y_i.
      2.2.2. חשבו את שגיאת המודל עם וקטור המשקלים הנוכחי w: $f(x_i, y_i, w)$.
      2.2.3. חשבו את וקטור הגרדיאנט $g(x_i, y_i, w) = \nabla_w f(x_i, y_i, w)$.
      2.2.4. עדכנו את וקטור המשקולות w:

$$w \Longleftarrow w - \eta_k g(x_i, y_i, w)$$

3. חזרו על שלב 2 עד שמתקיים <u>אחד</u> משני תנאי העצירה הבאים:
   - אין יותר עדכון של w (כלומר w הנוכחי זהה ל-w הקודם).
   - הושלמו K חזרות על סדרת האימון.

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:

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

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

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]):

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

$w = (w_1, w_2, \dots, w_D)$ 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:

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

[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 $w^T x + b$ לביטוי הכולל אך ורק מכפלה יחידה $w^T x$. שימו לב ששינוי הרישום לא משנה את העובדה שהפונקציה אפינית.

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

$a(x_i)$ 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 הסימן, שמותאמת *לסיווג בינארי* בין שתי מחלקות:

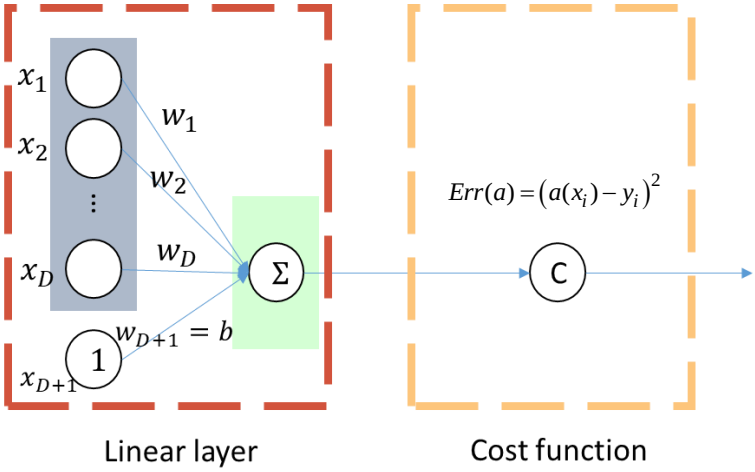
$$\hat{y} = sign(w^T x) = \begin{cases} -1, & w^T x < 0 \\ +1, & w^T x > 0 \end{cases}$$

כאשר מתקבל הערך 0 מסווגים אותו באופן שרירותי לאחת המחלקות.

נתבונן בפונקציה $a(x) = w^T x$. המשוואה $a(x) = 0$ מגדירה משטח החלטה המפריד בין שתי מחלקות: $a(x) < 0$ ו-$a(x) \geq 0$, 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

<u>אימון באמצעות Gradient Descent</u>

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

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

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

אלגוריתם אימון ADALINE

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

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

שלבי האימון:

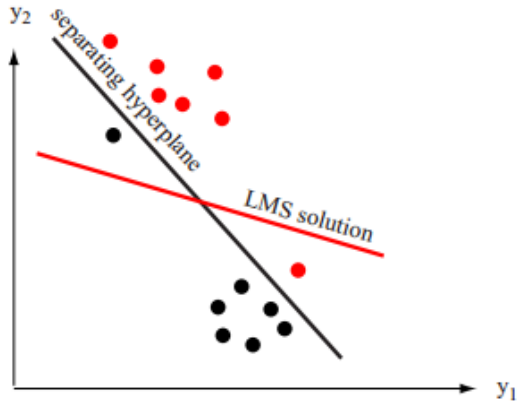
1. אתחול: אתחלו את הווקטור w בערך כלשהו.
2. עבור החזרה ה-k $(k = 1,2,3,\dots,K)$:
   2.1. קבעו את קצב האימון η_k עבור האיטרציה הנוכחית.
   2.2. עברו על דוגמאות סדרת האימון $i = 1,2,3,\dots,n$:
      2.2.1. קבלו את הווקטור $x_i \in \mathbb{R}^{D+1}$ והתיוג $y_i \in \{-1,+1\}$.
      2.2.2. חשבו את יציאת הרשת עם וקטור המשקלים הנוכחי w: $\hat{y}_i = w^T x_i$.
      2.2.3. עדכנו את וקטור המשקלים w:

$$w \leftarrow w - \eta_k x_i \left(\hat{y}_i - y_i\right) = w - \eta_k x_i \left(w^T x_i - y_i\right)$$

3. חזרו על שלב 2 עד שמתקיים אחד משני תנאי העצירה הבאים:
   - אין יותר עדכון של w (כלומר w הנוכחי זהה ל- w הקודם).
   - הושלמו K חזרות על סדרת האימון.

הערות:

- האלגוריתם לעיל הוא מקרה פרטי של האלגוריתם שראינו קודם לאימון רשתות.
- ישנן מספר דרכים לעדכן את גודל הצעד, כך שבכל איטרציה הוא יהיה קטן או שווה לגודל הצעד מהאיטרציה הקודמת. דרך פשוטה אפשרית היא לפי הנוסחה: $\eta_k = \eta_0/k$.
- כאשר נבחרת דרך פשוטה לעדכון גודל הצעד (כמו למשל בדוגמה הנ"ל), 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.

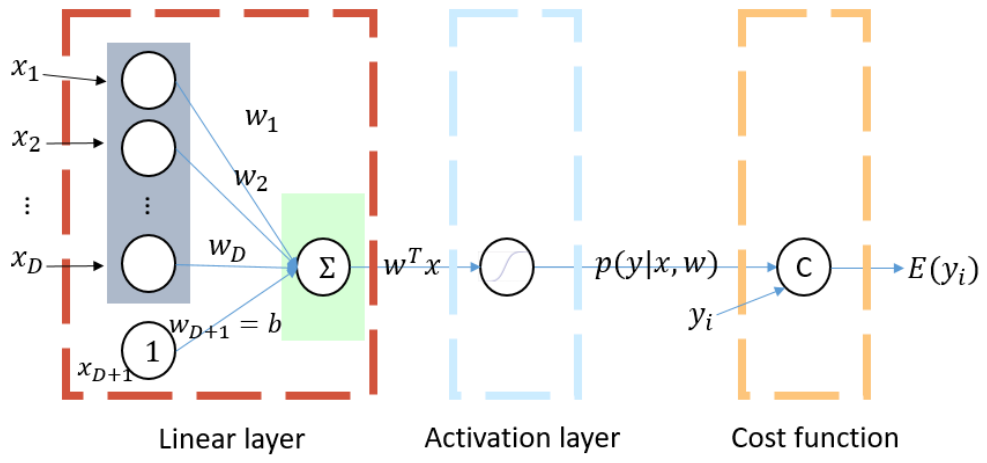
רגרסיה לוגיסטית הינו אלגוריתם סיווג בינארי שפותח בשנת 1958 ע"י David Cox.

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 $p(y \mid x, w)$.

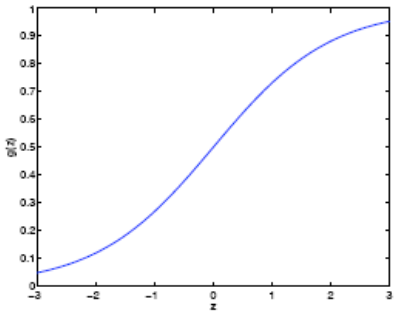
משיגים את הייצוג ההסתברותי ע״י שימוש בפונקציית הסיגמואיד :

$$p(y|x,w) = Ber\left(y\middle|\sigma(w^T x)\right)$$

כאשר הסיגמואיד מוגדר כך :

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

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 y_i יקבל ערך 1.

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

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

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:

$$NLL(x,y,w) = -(\mathbb{1}_0(y)log\pi_0 + \mathbb{1}_1(y)\log\ \pi_1) = -((1-y)log\pi_0 + y\log\ \pi_1)$$

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:

$$L(X,Y,w) = \frac{1}{n}\sum_{i=1}^{n} NLL(x_i,y_i,w) = -\frac{1}{n}\sum_{i=1}^{n}\left((1-y_i)log\left(1-\pi_{i,1}\right) + y_i\log\left(\pi_{i,1}\right)\right)$$

מהי הנגזרת של הביטוי הנ"ל לפי w?

$$\frac{\partial L}{\partial w} = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial w}NLL(x_i,y_i,w)$$

כדי לחשב ביטוי מפורש עבור דוגמה אחת, נשתמש בכלל השרשרת, כאשר נגדיר $a_i = w^T x_i$:

$$\frac{\partial}{\partial w}NLL(x_i,y_i,w) = \frac{\partial a_i}{\partial w}\frac{\partial \pi_{i,1}}{\partial a_i}\frac{\partial}{\partial \pi_{i,1}}NLL(x_i,y_i,w)$$

כעת נרשום את הביטויים בצורה מפורשת:

$$\frac{\partial L}{\partial \pi_{i,1}} = \frac{\partial}{\partial \pi_{i,1}}\left(\frac{1}{n}\sum_{j=1}^{n}NLL(x_j,y_j,w)\right) = \frac{1}{n}\sum_{j=1}^{n}\frac{\partial}{\partial \pi_{i,1}}NLL(x_j,y_j,w) = \frac{1}{n}\frac{\partial}{\partial \pi_{i,1}}NLL(x_i,y_i,w)$$

$$= \frac{1}{n}\left(\frac{1-y_i}{1-\pi_{i,1}} - \frac{y_i}{\pi_{i,1}}\right)$$

$$\frac{\partial \pi_{i,1}}{\partial a_i} = \sigma(a_i)\left(1-\sigma(a_i)\right)$$

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

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

שימו לב כי בשאלה 3 נבצע גזירה מטריצית השייכת ל<u>חדו"א מטריצית</u>. כללי הגזירה המטריציים מאוד דומים לאלו של סקלרים. במעבדה זו לא ניכנס להוכחתם.

המעוניינים בפיתוח המלא, יכולים למצוא הסבר תמציתי על הנושא ניתן למצוא <u>כאן</u>.

<u>אלגוריתם אימון רגרסיה לוגיסטית</u>

<u>קלט</u>: סדרת דוגמאות מתויגות לאימון $\{(x_i, y_i)\}_{i=1}^{n}$, פרמטר קצב האימון ההתחלתי $\eta_0 > 0$, מספר חזרות מרבי על סדרת האימון K.

<u>פלט</u>: וקטור משקולות $w = (w_1, w_2, ..., w_D, w_{D+1})$.

<u>שלבי האימון</u>:

1. אתחול: אתחלו את הווקטור w בערך כלשהו (למשל, $w = (1,1,...,1)$).
2. עבור החזרה ה-k $(k = 1,2,3, ..., K)$:
   2.1. קבעו את קצב האימון η_k עבור האיטרציה הנוכחית.
   2.2. עברו על דוגמאות סדרת האימון $i = 1,2,3 ..., n$:
      2.2.1. קבלו את הווקטור $x_i \in \mathbb{R}^{D+1}$ והתיוג $y_i \in \{0,1\}$.
      2.2.2. חשבו את שגיאת המודל עם וקטור המשקולות הנוכחי w: $NLL(x_i, y_i, w)$.
      2.2.3. עדכנו את וקטור המשקולות w בעזרת הנוסחאות מהעמוד הקודם:

$$w \longleftarrow w - \eta_k \frac{\partial}{\partial w} NLL(x_i, y_i, w)$$

3. חזרו על שלב 2 עד שמתקיים <u>אחד</u> משני תנאי העצירה הבאים:
   - אין יותר עדכון של w (כלומר w הנוכחי זהה ל-w הקודם).
   - הושלמו K חזרות על סדרת האימון.

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).

<u>רגרסיה לוגיסטית כמודל שכבות</u>

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:

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

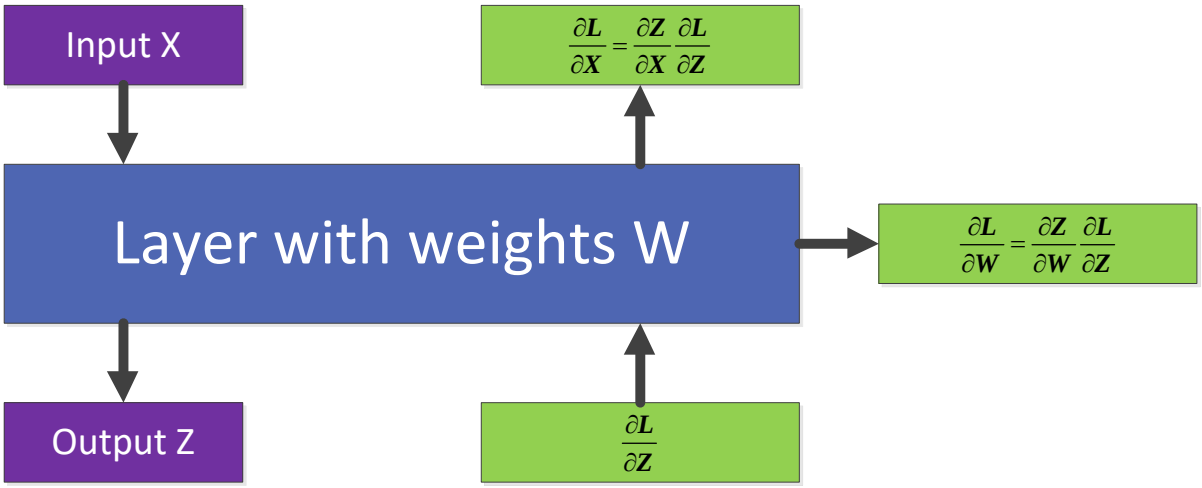
כאשר L היא פונקציית המחיר, σ היא הסיגמואיד ו-a היא הפונקציה האפינית.

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

$$g(x_i, y_i, w) = \frac{\partial L}{\partial w} = \frac{\partial a}{\partial w}\frac{\partial \sigma}{\partial a}\frac{\partial L}{\partial \sigma}$$

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, יציאות ונגזרות

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

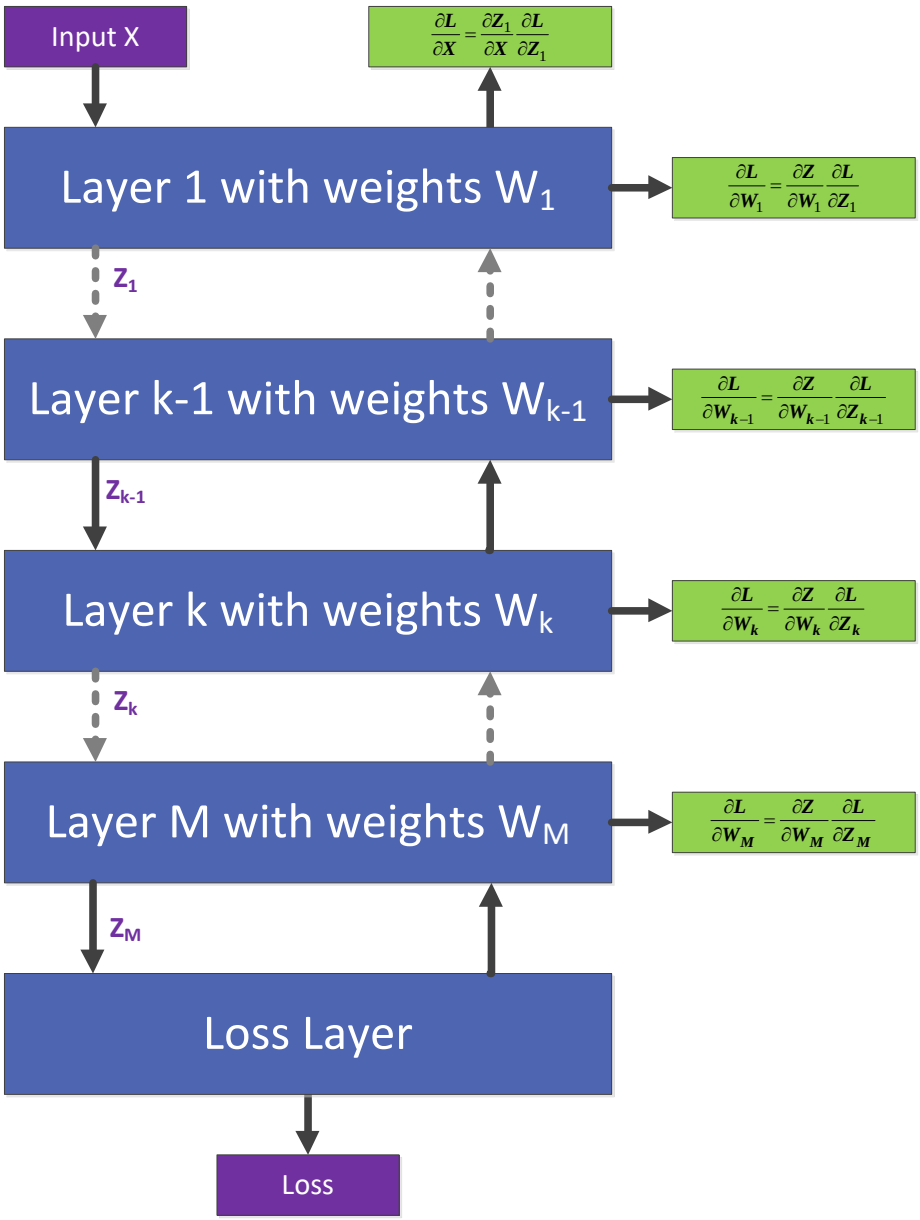
1. פונקציית מעבר קדמית.
2. נגזרת לפי פרמטרי השכבה.
3. נגזרת לפי כניסת השכבה.

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

שימו לב שבטבלה זו כל שכבה עומדת בפני עצמה ואינה תלויה באחרות.

	מעבר קדמי	נגזרת לפי פרמטרים	נגזרת לפי הכניסה
שכבה לינארית			
סיגמואיד		N/A	
NLL		N/A	

לאחר שחישבנו את הפונקציות לכל שכבה **בנפרד**, 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 בת M שכבות 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 $C > 2$ מחלקות, וקטור הסיווג הנתון יהיה וקטור בגודל C ויכיל אפסים בכל המיקומים פרט לאחד - באינדקס המציין את המחלקה אליה משתייכת הנקודה הרלוונטית. נרצה שמוצא הרשת יהיה וקטור בגודל C המכיל עבור כל אינדקס את ההסתברות של השתייכות הכניסה למחלקה באינדקס זה. כלומר, וקטור של C 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.

$$softmax(o_j) = \frac{e^{o_j}}{\sum_{k=1}^{C} e^{o_k}}$$

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 $\frac{\partial softmax(o)}{\partial o_j}$. נסמן $p_j = softmax(o_j)$ ונפריד למקרים:

$$\frac{\partial softmax(o_j)}{\partial o_j}, \qquad \frac{\partial softmax(o_{k\neq j})}{\partial o_j}$$

בשאלות ההכנה תוכיחו כי מתקיים:

$$\frac{\partial p_k}{\partial o_j} = \begin{cases} p_k(1-p_k), & k = j \\ -p_k p_j\ , & k \neq j \end{cases} = p_k(\delta_{kj} - p_j) \qquad \delta_{kj} = \begin{cases} 1, & k = j \\ 0, & k \neq j \end{cases}$$

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

עבור בחירה מתוך C מחלקות, פונקציית ה-NLL (עבור דוגמה אחת) נראית כך:

$$NLL(x, y, w) = -\sum_{k=1}^{C} y_k \log\ p_k$$

$$\Longrightarrow \frac{\partial}{\partial p_k} NLL(x, y, w) = -\frac{y_k}{p_k}$$

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

$$\frac{\partial}{\partial o_j} NLL(x, y, w) = -\sum_{k=1}^{C} y_k \frac{\partial \log(p_k)}{\partial o_j} = -\sum_{k=1}^{C} \frac{y_k}{p_k}\frac{\partial p_k}{\partial o_j} = -\sum_{k=1}^{C} \frac{y_k}{p_k} p_k(\delta_{kj} - p_j)$$

$$= -\sum_{k=1}^{C} y_k(\delta_{kj} - p_j) = -y_j + p_j \underbrace{\sum_{k=1}^{C} y_k}_{=1} = p_j - y_j$$

ניזכר כי פונקציית המחיר אותה אנחנו ממזערים היא:

$$L(X, Y, w) = \frac{1}{n}\sum_{i=1}^{n} NLL(x_i, y_i, w)$$

לכן עבור דוגמה אחת (x_i, y_i), נקבל את הביטוי הבא:

$$\frac{\partial L}{\partial o_{i,j}} = \frac{\partial}{\partial o_{i,j}}\left(\frac{1}{n}\sum_{r=1}^{n} NLL(x_r, y_r, w)\right) = \frac{1}{n}\sum_{r=1}^{n} \frac{\partial}{\partial o_{i,j}} NLL(x_r, y_r, w) = \frac{1}{n}\frac{\partial}{\partial o_{i,j}} NLL(x_i, y_i, w)$$

$$= \frac{1}{n}(p_{i,j} - y_{i,j})$$

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. הסבירו את תשובותיכם.
   א. זיהוי דובר מקטע אודיו.
   ב. חיזוי שער מניה לאחר מספר ימים.
   ג. זיהוי טקסט מתמונה.
   ד. מסנן דואר זבל (Spam).
   ה. גילוי מספר אנשים שנמצא בתמונה.
   ו. 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 פונקציות ה-backward שלהן (גזירה של הפונקציה לפי הקלט ולפי הפרמטר):

1. $f(x,a) = \ln(ax)$
2. $f(x,a) = (x-a)^2$
3. $f(x,a,b) = \max(b, ax)$
4. $f(x,a,b) = \sqrt{ax+b}$
5. $f(x) = \tanh(x)$

הניחו כי הפונקציות והמשתנים הינם סקלרים.

3. הראו כי הנגזרת של פונקציית הסיגמואיד $\sigma(z)$ נתונה ע"י הביטוי:

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

4. תהי $f: \mathbb{R}^n \to \mathbb{R}$ פונקציה דיפרנציאבילית. נגדיר $y = A^T x + b$ כאשר $b \in \mathbb{R}^n, A \in \mathbb{R}^{m\times n}$ הינם מטריצה ווקטור ידועים. בטאו את הנגזרות הבאות של $f(y) = f(A^T x + b)$ באמצעות הנגזרת $\frac{\partial f}{\partial y} \in \mathbb{R}^n$ והמשתנים b, A, x בלבד. עבור כל סעיף יש לציין את הממד הצפוי של הנגזרת, ולהראות שאכן הוא מתקבל.

הדרכה: בסעיפים א' ו-ב' יש להשתמש בכלל השרשרת למקרה הווקטורי. בסעיף ג' הגזירה היא לפי מטריצה ולא לפי וקטור, ולכן יש לגזור את הביטוי איבר-איבר לפי איברי המטריצה A. הביאו את כל הביטויים לייצוג מטריצי.

   א. $\frac{\partial f}{\partial x}$ - נגזרת של f לפי הוקטור x.
   ב. $\frac{\partial f}{\partial b}$ - נגזרת של f לפי הוקטור b.
   ג. $\frac{\partial f}{\partial A}$ - נגזרת של f לפי המטריצה A.

כעת נסתכל על אוסף דוגמאות $\{x_i\}_{i=1}^{K}$ שעבורם מחושבת הנגזרת במקביל (Batch). לצורך כך נגדיר $y_i = A^T x_i + b$, ואת המטריצות $X = [x_1, x_2, \cdots, x_K] \in \mathbb{R}^{m\times K}$ ו- $Y = [y_1, y_2, \cdots, y_K] \in \mathbb{R}^{n\times K}$.
במקרה זה, הפונקציה שאותה נגזור הינה $F = \sum_{i=1}^{K} f(y_i) = \sum_{i=1}^{K} f(A^T x_i + b)$. בטאו את הנגזרות הבאות של F באמצעות הנגזרת $\frac{\partial F}{\partial Y} \in \mathbb{R}^{n\times K}$, המשתנים b, A, X ווקטור האחדות $\mathbf{1}$ בלבד.

הדרכה: היעזרו בלינאריות של הנגזרת ובתוצאות סעיפים א'-ג', והביאו את הביטויים לייצוג מטריצי.

הערה: ניתן לפתור את הסעיפים הבאים גם ללא שימוש בטנזורים.

ד. $\frac{\partial F}{\partial X}$ - נגזרת של F לפי המטריצה X.

ה. $\frac{\partial F}{\partial b}$ - נגזרת של F לפי הווקטור b.

ו. $\frac{\partial F}{\partial A}$ - נגזרת של F לפי המטריצה A.

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

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

6. בהינתן הפונקציה $f(x) = x^2$ ונקודת התחלה $x_0 = -1$.

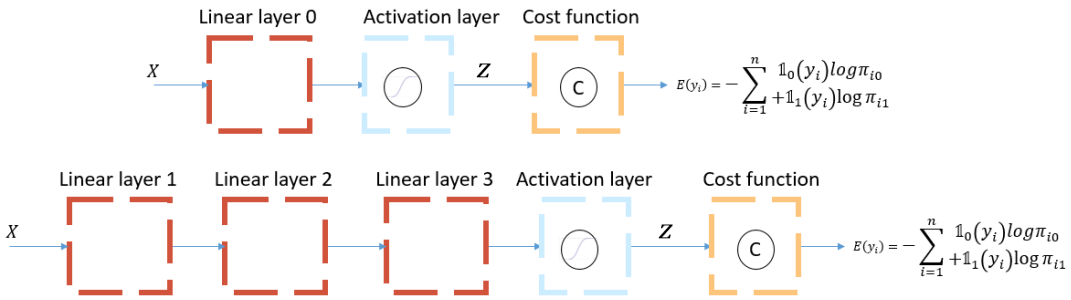
א. חשבו את שלושת האיטרציות הראשונות של gradient descent עם גודל צעד $\eta = 0.1$.

ב. חשבו את שלושת האיטרציות הראשונות של gradient descent עם גודל צעד $\eta = 3$.

ג. מהי חשיבותו של בחירת גודל הצעד בתהליך האימון?

ד. מהו גודל הצעד המרבי שעבורו gradient descent יתכנס לנקודת המינימום? הוכיחו.

7. האם הרשתות הבאות שקולות? הסבירו את תשובתכם.



הערה: שתי רשתות תחשבנה לשקולות אם כל פונקציה שרשת אחת יכולה לממש (מכניסה x למוצא z), גם הרשת השניה יכולה לממש, והפוך.

8. צרפו את הטבלה מפרק "רגרסיה לוגיסטית כמודל שכבות" בה מילאתם את הפונקציות של כל שכבה.

9. צרפו לדו"ח כצילום מסך (ברור) את החלקים שהשלמתם מתוך הקוד עבור סיווג האירוסים. אין צורך להגיש את הקוד עצמו.

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 במהלך ביצוע הניסוי.

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

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

דרישות מחשוב:

- MATLAB 2020a לפחות
  - Deep Learning Toolbox
  - Parallel Computing Toolbox
  - Statistics and Machine Learning Toolbox
  - AlexNet pre-trained network

מומלץ **לא להשתמש** בלולאות במימוש הפונקציות במהלך הניסוי. מימוש בלולאות הינו איטי יותר ואינו משתמש בספריות של חישוב מקבילי. כל הפונקציות בניסוי ניתנות למימוש יעיל ללא שימוש בלולאות כלל.

חלק א' – רגרסיה לוגיסטית

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

נזכיר, כי שכבות של רשתות נוירונים מאמנים באמצעות שתי פונקציות:

- פונקציית מעבר קדמית (שלב 2.3 באלגוריתם).
- פונקציית מעבר אחורית (שלב 2.4 באלגוריתם).

את הפונקציות האלו נממש בעזרת הטבלה שחישבתם בשאלה 8 בהכנה.

1. נתון מידע המסודר במטריצה בגודל MxN, ונתונה שכבה ברשת שמקבלת את המידע בכניסה כאשר מבוצע ברשת מעבר קדמי. רשמו כיצד משפיעים ממדי הכניסה על ממדי המוצא מהשכבה עבור שכבה לינארית, שכבת סיגמואיד ושכבת NLL.

2. כעת מבצעים באותה הרשת מעבר אחורי. עבור אותן שכבות שמופיעות בסעיף הקודם, כיצד ניתן לדעת מהם ממדי הגרדיאנטים המועברים אחורה לשכבה הקודמת ולעדכון המשקולות? הסבירו.

הסתכלו בקבצי ה-MATLAB הנתונים לכם ופתחו את התיקייה part1 בתוך lab1. שימו לב לתיקיית layers בה נמצאות פונקציות המעבר שיש לממש.

דגשים חשובים שצריך לקרוא לפני שמתחילים לממש:

- את השכבות יש לממש בעזרת הטבלה שהשלמתם בשאלה 8 בהכנה.
- כל פונקציית backward צריכה להחזיר את הגרדיאנט המלא הנוכחי, כלומר הגרדיאנט המחושב בצעד הנוכחי כפול הגרדיאנט שחושב עד כה (לפי כלל השרשרת).
- עבור כל פונקציה, על המימוש שלכם לתמוך בכניסה יחידה (x שהוא וקטור עמודה) וגם במספר כניסות המתוארות ע"י מטריצה שכל עמודה בה היא דוגמה נפרדת.
- **השתמשו בקבצי הבדיקה המצורפים בתת התיקייה tests תחת תיקיית layers:**
  - לכל שכבה הריצו את קובץ הבדיקה המתאים לה, למשל test_affine.
  - 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 עליה.

3. השלימו את הקוד והריצו טסטים עבור הפונקציות affine_forward, binary_nll_loss_forward, sigmoid_forward, affine_backward, binary_nll_loss_backward, sigmoid_backward. ודאו כי אכן מתקבלים הממדים להם ציפיתם בסעיפים הקודמים.

4. עבור רגרסיה לוגיסטית כמודל שכבות, מהו סדר השימוש הנכון בפונקציות שממשתם? התייחסו הן למעבר הקדמי והן למעבר האחורי כפי שמופיע ברקע התאורטי.

5. כיצד אתם מצפים שיראה המפריד בין המחלקות עבור הרגרסיה הלוגיסטית? נמקו.

6. השלימו את הקוד עבור הפונקציה logisticRegression.

7. הריצו את IrisClassification שלוש פעמים, פעם אחת עם קצב התכנסות 0.02, פעם שניה עם קצב התכנסות 0.003, ופעם שלישית עם קצב התכנסות 2. צפו בתהליך ההתכנסות. הוסיפו לדו"ח המסכם שלכם את פלטי הריצות.
ציינו את ההבדלים בין שלושת ההרצות שביצעתם והסבירו מה קורה בכל אחת מההרצות ולמה?

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

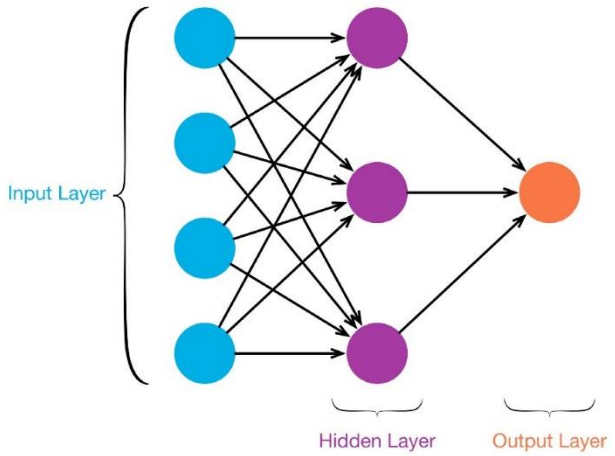
כעת נעבור לבעיה יותר מורכבת: זיהוי ספרות כתב יד.

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 MNISTClassification בתיקיית part2 והריצו את שני המקטעים הראשונים. התבוננו בתמונות לדוגמה מהמאגר.

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 סיגמואיד, בשכבה השנייה נשתמש באקטיבציה מסוג softmax ופונקציית ההפסד תהיה 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 סיגמואיד
   f. $softmax(x)$ פונקציית האקטיבציה softmax
   g. $l(\hat{y}, y)$ פונקציית ההפסד

   אילו ממדים מושפעים מרוחב השכבה הנסתרת בביטויים הנ״ל?

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 ה-NLL.

2. פתחו את התיקייה layers והשלימו את הקוד עבור הפונקציות הבאות לפי הנוסחאות המוצגות בחלק עבור סיווג מתוך מחלקות רבות:
   - multi_nll_loss_forward.m
   - softmax_forward.m
   - nll_and_softmax_backward.m

   גם כאן תקפים אותם הדגשים מחלק א׳. השתמשו בקבצי הבדיקה 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 (*). כאשר מבצעים חלוקת וקטורים, **תמיד** יש להשתמש ב-(./) כאשר הממדים של המונה והמכנה אינם זהים.

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

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

3. השלימו את השורות הנחוצות ב-trainTwoLayerPerceptron.
4. היעזרו בשכבות שמימשתם בשני החלקים של הניסוי עד כה, והשלימו את הקוד ב-forwardPass וב-backwardPass. שימוש בשכבות שעברו את הבדיקות יקטין את הסיכוי לטעויות בריצה עצמה.
5. הריצו את יתר המקטעים MNISTClassification. ודאו כי המודל מתכנס, וצרפו את התוצאות הסופיות והגרף המתקבל.
6. הסבירו כיצד גודל השכבה הנסתרת משפיע על המודל. התייחסו בתשובתכם לסיבוכיות המודל ושיקולי ביצועים (מקום, זמן, דיוק). הסתכלו באיור 5 ואיור 6 לקבלת אינטואיציה.
7. בחלק זה האימון מתבצע בעזרת מקבצים (batches), לעומת החלק הראשון בו האימון התבצע דוגמה אחר דוגמה. מהם היתרונות והחסרונות של אימון בעזרת מקבצים? הסבירו.

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 סינטטי דומה ל-MNIST בשם Digits, המכיל תמונות של ספרות ותיוגים שהם הספרות שבתמונות.

❖ קראו את נספח א' – Matlab Layers חלק א', והבינו את מטרתה של כל שכבה. אנו נשתמש בשכבות המתוארות בנספח זה.

בניגוד לחלק הקודם, האקטיבציה הראשונה בה נשתמש היא ReLU, ולא סיגמואיד. שכבה זו מופיעה בנספח ב'. בחלק ההכנה למפגש הבא נלמד מה ההבדל ביניהם ומה היתרונות של ReLU. השימוש בשכבת ה-ReLU זהה לשימוש שעשינו בסיגמואיד. האקטיבציה השנייה שנשתמש בה תהיה softmax.

1. בקובץ DigitsClassification, החליפו את השורות הקיימות בשכבות הרצויות והפרמטרים שלהן לפי המודל מהחלק הקודם עם השינויים שצוינו בפסקה הקודמת.
   רשימת השכבות הרלוונטיות לניסוי זה נמצאת בנספח א' – Matlab Layers חלק א'. שימו לב לפרמטרים של כל שכבה.
2. הסבירו מה המשמעות של כל אחד מהיפר-הפרמטרים המופיעים ב-hparams בקובץ DigitsClassification.

בחרו בגודל של 400 עבור השכבה הנסתרת והריצו את DigitsClassification. שימו לב לתדפיס ב- command window של MATLAB:

Epoch	Iteration	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss

הפרמטרים אשר יעניינו אותנו בעיקר בחלק זה הם ה-Loss וה-Accuracy, של ה-train למול ה-Validation.

ניתן לראות את ה-Loss וה-Accuracy של סט האימון בגרף ההתכנסות אשר מוצג בתהליך האימון. שימו לב שמוצגים בקו מקווקו נתוני ה-validation.

בכל ריצה שנבצע יש להוסיף את גרף ההתכנסות ואת הטבלה הנ״ל לדו״ח. אין צורך להעתיק את הטבלה כולה, מספיק להוסיף רק את 5-10 השורות האחרונות בה. זכרו לרשום את ה-Loss וה-accuracy הסופיים של ה-train וה-validation.

3. שנו את מספר האפוקים ל-10 והשוו הרצות עם גדלי צעד שונים: 3e-3, 1e-3, 8e-4, 5e-4. הסתכלו על התכנסות ה-Loss וה-Accuracy לאורך האימון.
מתי מתקבלת התוצאה הטובה ביותר? הסבירו את התוצאות.
4. בחרו בגודל הצעד המיטבי והשוו הרצות עם גדלי שכבה נסתרת שונים: 200, 500, 700, 1000. הסתכלו על התכנסות ה-Loss וה-Accuracy לאורך האימון.
מתי מתקבלת התוצאה הטובה ביותר? הסבירו את התוצאות.
5. בחרו בגודל השכבה הנסתרת המיטבי והריצו את המודל עם גודל צעד של 5e-5. הסתכלו על התכנסות ה-Loss וה-Accuracy לאורך האימון. מה הסיבה להבדלים ב-Loss וב-accuracy 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.
6. כעת, הריצו את המודל עם גודל צעד של 0.5. מה קרה לגרף ההתכנסות? הסבירו מדוע.
7. עבור בעיות סיווג, הסבירו את ההבדל בין ערך פונקציית המחיר (הגרף האדום) לבין אחוז הדיוק בסיווג (הגרף הכחול). כיצד פונקציית המחיר עוזרת לנו להביא את שגיאת הסיווג למינימום?

רקע למעבדה – מפגש שני

הקדמה

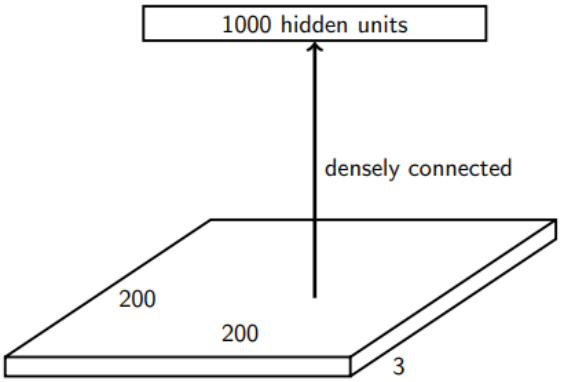
בחלק השני של הניסוי נכיר שכבות נוספות המהוות אבני בסיס בלמידה עמוקה. בפרט, נכיר שכבה הנקראת "שכבת קונבולוציה". לשכבה זו חשיבות גדולה בלמידה עמוקה והיא אפשרה לתחום לגדול לאן שהוא נמצא היום. כמו כן, נלמד על שכבות לא ליניאריות, גישות אופטימיזציה מתקדמות, רגולריזציה ושיקולים בתכנון רשתות.

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.

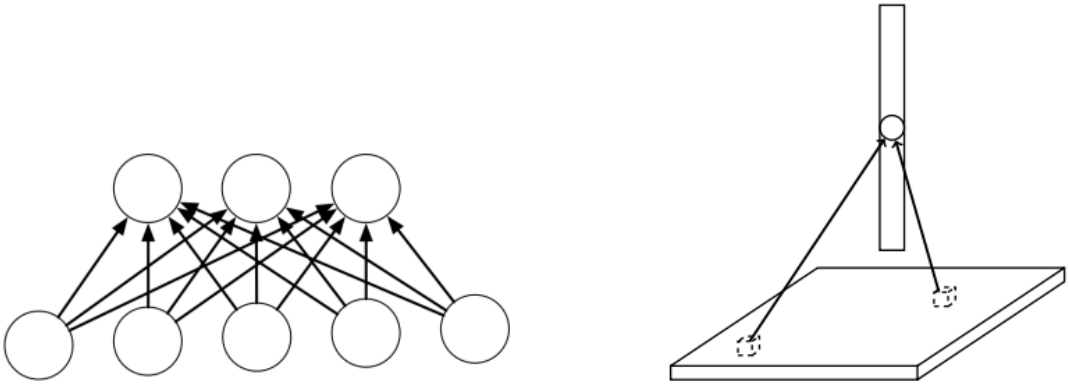
בעיה נוספת: מה יקרה אם נצלם את (כמעט) אותה התמונה רק בהיסט קטן ימינה? שימו לב שבמקרה זה כל פיקסל יוכפל בפרמטר אחר!

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

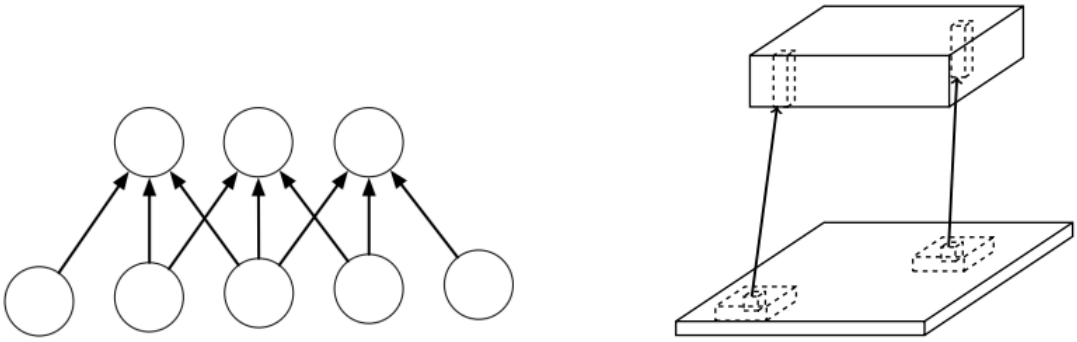
סכמתית נתאר רשתות לינאריות כך:



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.

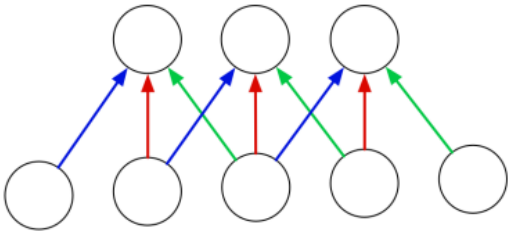
שכבות לינאריות לעיתים מכונות גם שכבות fully-connected, כלומר בעלות חיבוריות מלאה.

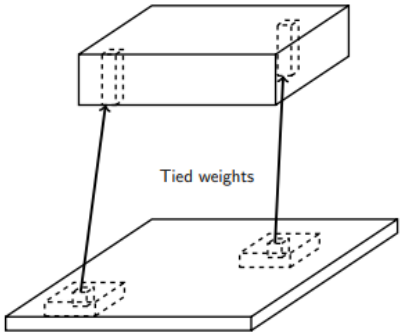
כעת נשנה מעט את השכבה. נרצה שמוצא השכבה תהיה גם היא מטריצה תלת-ממדית (גובה, רוחב ועומק. העומק במקרה של תמונת צבע הוא 3). כמו כן, נדרוש שכל פיקסל יושפע רק מסביבה קטנה בתמונת הכניסה. השכבה הנ"ל תיראה כך:



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:

$$(f \star h)[m,n] = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} f[i,j]h[i+m, j+n]$$

<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.

1	1	1	0	0
0	1	1	1	0
0	0	$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$
0	0	$1_{\times 0}$	$1_{\times 1}$	$0_{\times 0}$
0	1	$1_{\times 1}$	$0_{\times 0}$	$0_{\times 1}$

Image

4	3	4
2	4	3
2	3	4

Convolved Feature

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

---

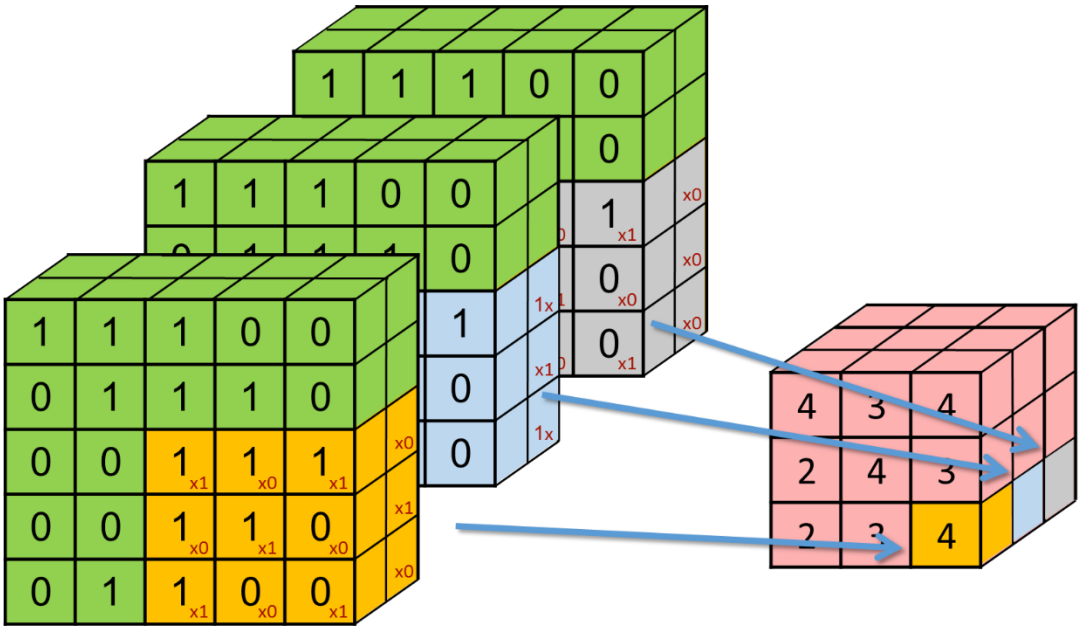
[4] זו היא כמעט אותה פעולה. בספרות של עיבוד אותות נהוג להגדיר את הקונבולוציה באמצעות שיקוף של אחת הפונקציות. הפעולה אותה אנו מציגים כאן היא ללא ההיפוך ונקראת פעולת *קרוס-קורלציה*. אנחנו ניצמד למונח "קונבולוציה" מכיוון שזהו המונח המקובל בתחום.

[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 ליניאריות, גם כאן נהוג להוסיף גודל קבוע לכל גרעין המכונה bias.



הנוסחה שמתארת את תהליך הקונבולוציה במטריצות תלת-ממדיות הינה:

$$(f \star h_p)[m,n,p] = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\sum_{k=0}^{K-1} f[i,j,k]h_p[i+m,j+n,k]$$

כאשר h_p הוא הגרעין ה-p.

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 קונובולוציה.

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

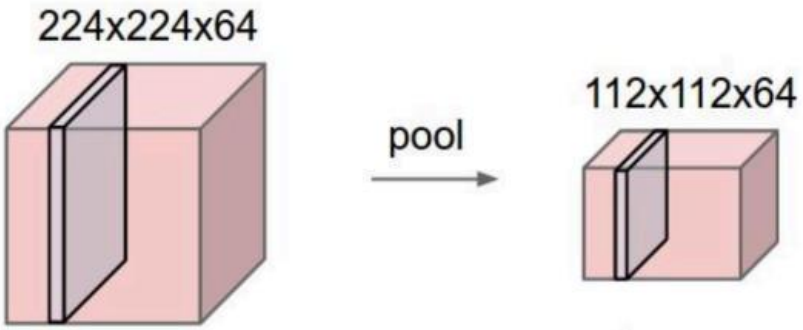
לשכבה זו 4 היפר-פרמטרים (פרמטרים שנקבעים מראש ולא נלמדים) ולא כוללת פרמטרים נלמדים כלל.

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 :



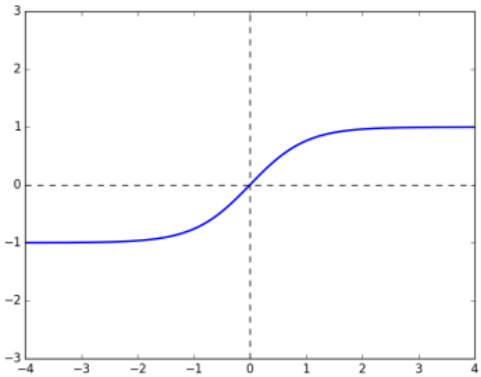
איור 16 – שכבת pooling על טנזור תלת מימדי

סוגי שכבות לא לינאריות

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

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

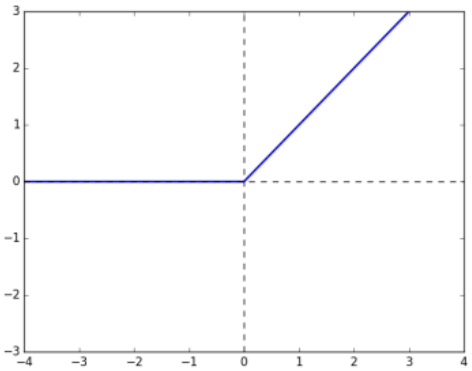
$$\frac{\partial\sigma}{\partial z}(0) = \sigma(0)\big(1 - \sigma(0)\big) = \frac{1}{2}\cdot\left(1 - \frac{1}{2}\right) = \frac{1}{4} < 1$$

כלומר, ככל שהרשת עמוקה יותר, כך נכפול את הגרדיאנט בעוד ערכים שקטנים מ-1 והשגיאה תדעך יותר.

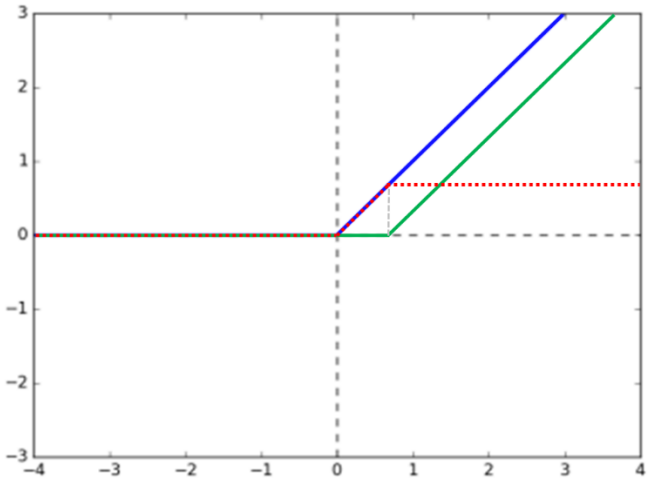
לבעיה הנ"ל ישנן מספר פתרונות. דרך אחת שפותרת את הבעיה היא להשתמש בשכבה לא לינארית אחרת.

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.

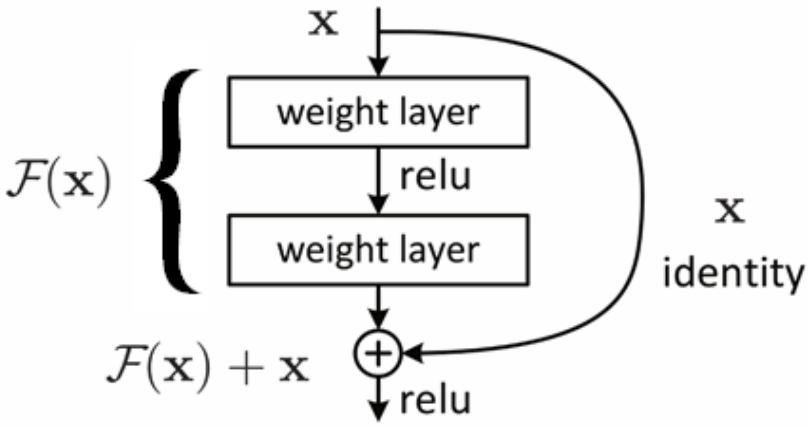


בכחול ובירוק ניתן לראות שני ReLU שונים, כאשר ה-ReLU הירוק מוזז. הזזה כזו יכולה להתקבל כאשר יש איבר הטיה בשכבה הקודמת (תזכורת: אם מדובר בשכבה לינארית המיוצגת ע"י $y = Ax + b$ אזי b הוא גורם ההטיה). באדום מצויר ההפרש בין הקו הכחול לקו הירוק. הפרש כזה יכול להיווצר כמכפלה וסכום של שכבה עוקבת בה המשקולות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

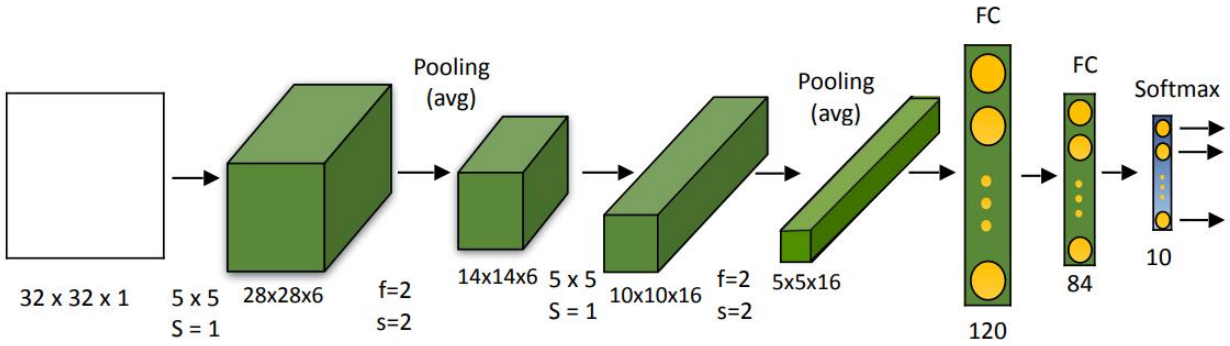
מוטיבציה אחת לשימוש בשכבות מסוג זה היא שהגרדיאנט של השכבה הוא:

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

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 (כאשר התמונות מרופדות באפסים), 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. שכבה 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. שכבה 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.

שכבת נרמול Batch Normalization

שכבת Batch Normalization (BN) היא שכבה שמטרתה לנרמל מקבצי דוגמאות (Batch) הנלמדים ברשת בשלב האימון, ע״י העברתם לממוצע אפס ושונות אחד. הנרמול מתבצע לכל אחד מערוצי המידע בנפרד. המוטיבציה לשימוש בשכבה היא תופעת ה-internal covariate shift, שגורמת לשינוי הסטטיסטיקה של המידע במעבר דרך שכבות הרשת, עקב האקראיות באתחול המשקולות בשכבות ואקראיות המידע. כאשר הסטטיסטיקה משתנה ממקבץ דוגמאות אחד למשנהו, השכבות נדרשות להתאים את עצמן בתהליך האימון למגוון סטטיסטיקות שונות, דבר הפוגע בביצועי הלמידה. שכבת BN מנרמלת את המידע תמיד לאותו מומנט ראשון ושני, ובכך מצמצמת מאוד בעיה זו.

כאשר מקבץ דוגמאות בגודל m עובר בשכבה זו, מחושבים המומנטים שלו לפי:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}\left(x_i - \mu_B\right)^2$$

לאחר חישוב המומנטים, מבוצעת פעולת הנרמול לכל ערוץ בנפרד, לפי:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

כאשר ε הוא קבוע קטן לצורך יציבות נומרית. לאחר שלב הנרמול מבצעים טרנספורמציה למידע המנורמל לצורך שמירת יכולת הייצוג של הרשת:

$$y_i = \gamma \hat{x}_i + \beta$$

כאשר הפרמטרים γ, β הם פרמטרים נלמדים במהלך תהליך האימון. גם פעולת הטרנספורמציה מתבצעת לכל ערוץ בנפרד. מוצא השכבה y מועבר ליתר השכבות לפי מבנה הרשת.

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 נגד כיוון הגרדיאנט כפול גודל הצעד.

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

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

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

1. <u>מומנטום</u>

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

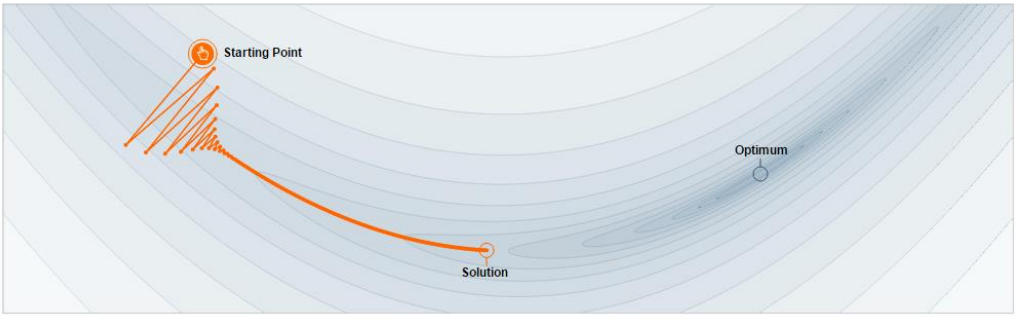
$$v_{k+1} = mv_k + \nabla f(x_k)$$

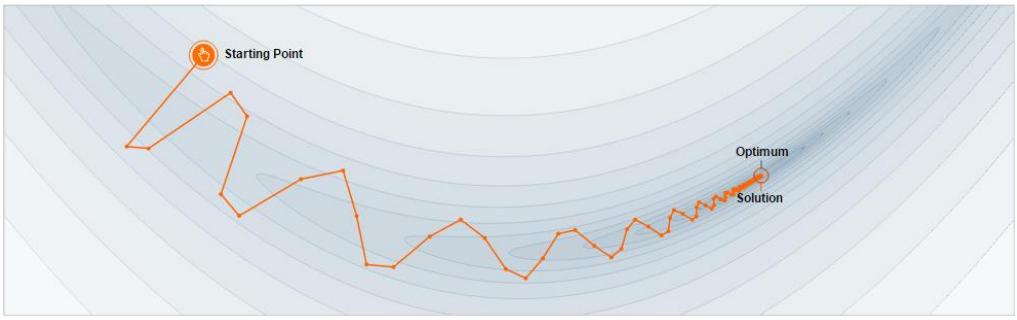
$$x_{k+1} = x_k - \eta v_{k+1}$$

האיבר m 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 כוח 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 אימון משלו הנקבע על סמך היסטוריית הגרדיאנטים שלו. אלגוריתם זה נותן קצב אימון גבוה לפרמטרים שמשתנים לעיתים רחוקות וקצב אימון 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.

שימוש ב-Scheduler

אימון מעשי של רשת לאורך מספר רב של אפוקים מחייב התאמה של גודל הצעד בהתאם להתקדמות בתהליך האימון. כאשר תהליך האימון אינו משתפר למשך מספר אפוקים, ניתן להקטין את גודל הצעד ולהשיג שיפור נוסף בביצועים. פעולה זו יכולה להתרחש מספר פעמים במהלך האימון, בהתאם לכמות האפוקים, אופי המידע ומבנה הרשת.

תהליך כזה, אשר משנה את גודל הצעד במהלך האימון, נקרא Scheduler. ישנן מספר דרכים שונות להשפיע על גודל הצעד במהלך האימון, ובהן שינוי גודל הצעד כל מספר אפוקים קבוע, שינוי בזמנים ידועים מראש או שינוי כאשר אין שיפור באחוזי האימון למשך זמן מסוים. קיימות גם שיטות שונות לביצוע השינוי עצמו, ובהן שינוי ע"י חיסור ערך קבוע, שינוי לינארי ע"י כפל בקבוע, שינוי אקספוננציאלי ועוד. בניסוי זה אנו משתמשים בשינוי לינארי של גודל הצעד כל זמן קבוע של אפוקים, וניתן להשפיע על מהלך האימון ע"י קביעת שני פרמטרים מתאימים (לחילופין ניתן גם לאמן ללא Scheduler, אבל זה לא מומלץ). שני הפרמטרים הללו נקראים 'LearnRateDropPeriod', המציין את מספר האפוקים בין פעולות שינוי של גודל הצעד, ו-'LearnRateDropFactor' , המציין את הקבוע הלינארי בו נכפל גודל הצעד בכל שינוי כזה. ישנם מספר כללי אצבע פשוטים לשימוש ב-Scheduler מסוג זה. תחילה, בכל שינוי של גודל הצעד מומלץ לשנות אותו בצורה ניכרת, ולא באחוזים בודדים. שנית, מומלץ להתאים את מספר האפוקים בין פעולות שינוי למספר האפוקים הכללי ולהתנהגות תהליך האימון. שינוי גודל הצעד באופן משמעותי לפני

שהאימון מספיק להתייצב (אפילו חלקית) עלול לפגוע בתוצאות. גם מספר השינויים שמתרחשים במהלך האימון יכול להשפיע על התוצאות הסופיות. ניתן לקרוא עוד על נושא זה כאן.

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.

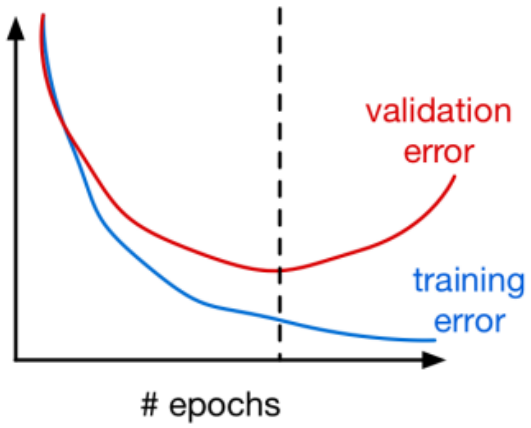
1. Early Stopping

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 כברירת מחדל. השיטה בה הם פועלים היא בדיקה של המגמה של סט הווליד‌ציה- 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:

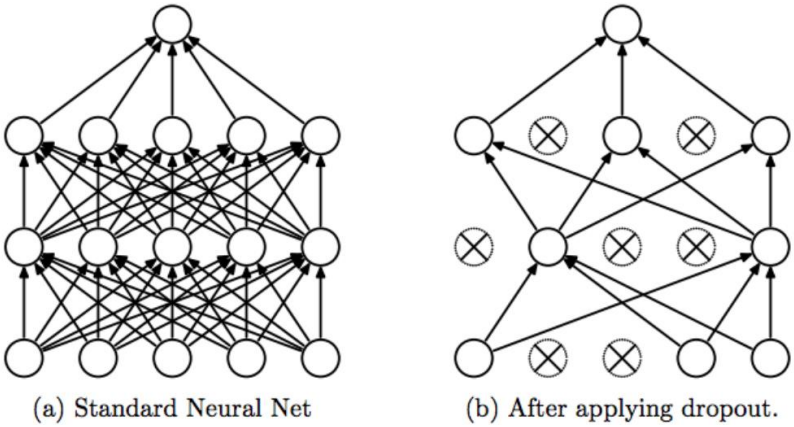
$$w^{t+1} = w^t - \eta_t\left(x\left(w^{t^T}x - y\right) - \alpha w\right) = (1-\eta_t\alpha)w - \eta_t\left(w^{t^T}x - y\right)$$

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. <u>Data Augmentation</u>

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 בתמונות: הזזה, שיקוף, סיבוב, שינוי גודל, שינוי בהירות והוספת רעש.

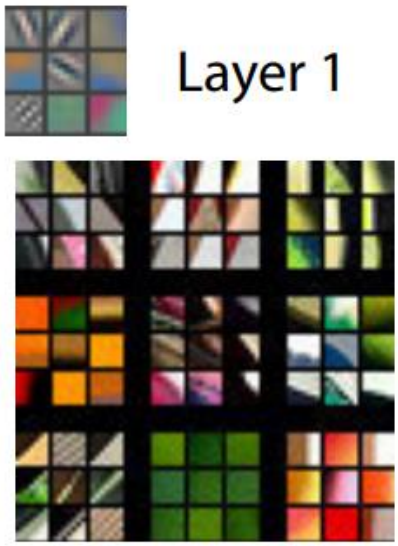
שימו לב! שימוש ב-Data augmentation אינו בהכרח משפר את התוצאות. הוספת מניפולציות רבות לתמונות עלולה להקשות על הרשת למצוא את המודל המאפיין אותן וכתוצאה מכך לפגוע בביצועים. גישה בריאה לשימוש ב-Data augmentation היא הדרגתית, ע״י הוספת מניפולציה אחת בכל פעם ובחינת ההשפעה על התוצאות. מומלץ להתחיל מזו שהינה ההגיונית ביותר בהתאם לאופי המידע, ובמידה ומושג שיפור להוסיף מניפולציה נוספת וכך הלאה. באותו האופן מומלץ להתייחס לפרמטרי המניפולציות, ולנקוט בגישה שמרנית.

מה נלמד בשכבות הפנימיות ברשת

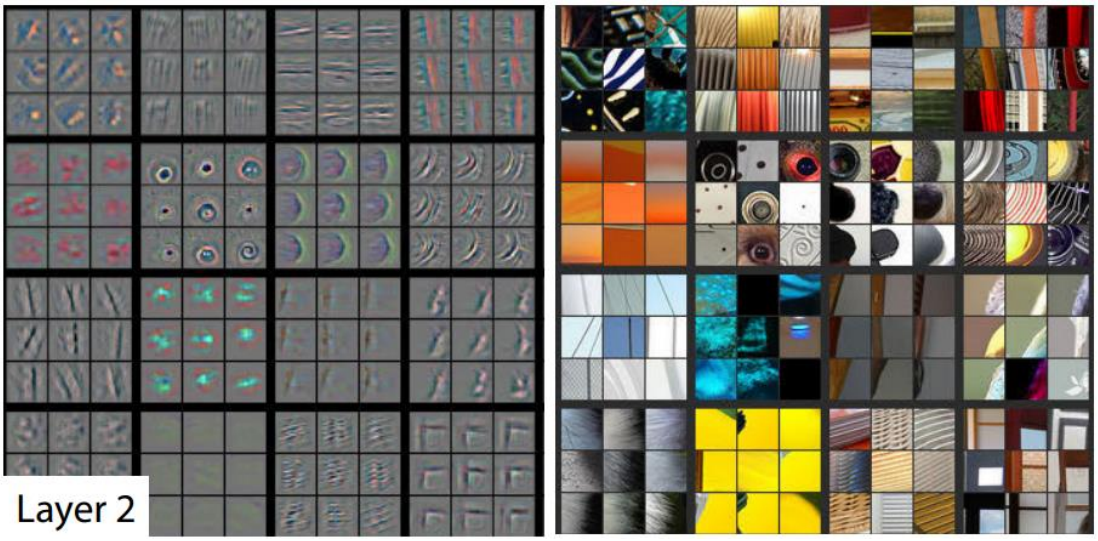
מתוך ניסיון להבין כיצד רשתות נוירונים עמוקות מצליחות לפתור בעיות מורכבות (בעיקר בעיות תמונה), החלו לחקור מה נלמד בכל שכבה ברשת .

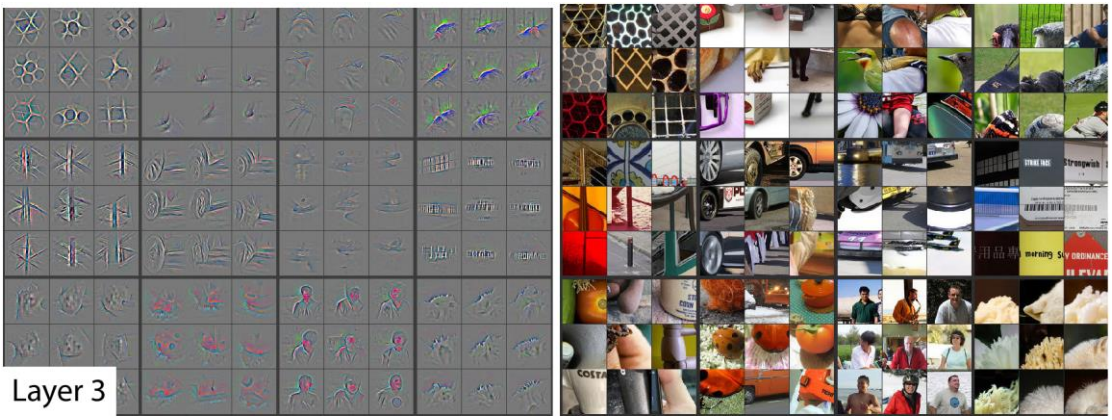
בתהליך זה גילו כי שכבות נמוכות לומדות לזהות מאפיינים בסיסים של תמונות כמו שפות ופינות, שכבות אמצעיות מזהות טקסטורות וצורות גאומטריות ושכבות גבוהות יותר מזהות עצמים מורכבים יותר כמו עיניים, פרצופים של בני אדם ושל בע״ח.

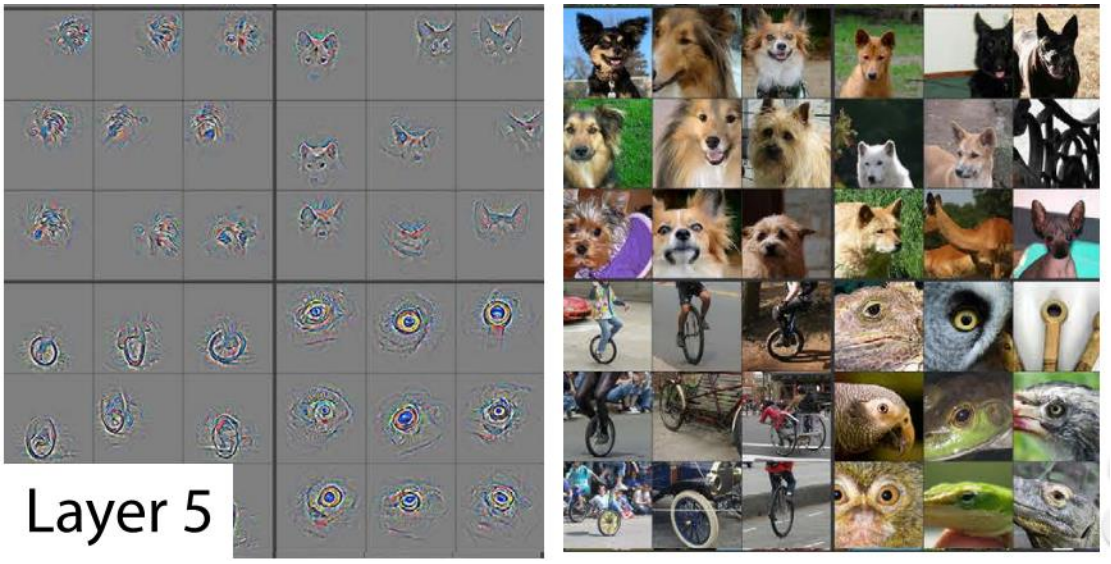
ויזואליזציה של שכבות בעומקים שונים ברשת שאומנה על סיווג תמונות ניתן לראות באיור 19. מימין רואים את תמונות המקור שהוזנו לרשת ומשמאל את האקטיבציות (פלט של שכבה ברשת לאחר השכבה הלא לינארית) של שכבות בעומקים שונים ברשת.



איור 19 – תמונות מייצגות עבור אקטיבציות של שכבה ראשונה ברשת סיווג. בעמוד הבא, משמאל תמונות מייצגות עבור אקטיבציות בשכבות עמוקות יותר, כתגובה לתמונות המקבילות להן מימין.







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 יישארו ללא שינוי, ולא יעברו אימון נוסף ע"י הדוגמאות החדשות (יישארו "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 ממדים $h_{in},\ w_{in},\ c_{in}$.
בנוסף, נתונה שכבת קונבולוציה בעלת הנתונים הבאים:
   - מסננים בגודל $k_h,\ k_w$.
   - מספר המסננים הוא d.
   - אין ריפוד או דילוג.
   - לתוצאת הקונבולוציה מתווסף bias.

   נסמן את ממדי המוצא של השכבה על ידי $h_{out},\ w_{out},\ c_{out}$.
   א. כתבו ביטויים ל- $h_{out},\ w_{out},\ c_{out}$ כפונקציה של הפרמטרים לעיל.
   ב. כמה פרמטרים נלמדים קיימים בשכבה זו?
   ג. אם נחליף את שכבת הקונבולוציה בשכבת fully-connected עם אותם ממדי מוצא, כמה פרמטרים נלמדים יהיו בה?
2. נתונה הפונקציה $f(x) = x^2$ ונקודת התחלה $x_0 = -1$, הדגימו שלוש איטרציות של gradient descent עם מומנטום $(m = 0.9)$.
   א. עבור גודל צעד $\eta = 0.01$.
   ב. עבור גודל צעד $\eta = 2$.
3. באיזה אופן משפיעה רגולריזציית L2 על המשקולות? התייחסו לפרמטר α בערכי קיצון ($\alpha \to 0$, $\alpha \to \infty$).
4. ציינו שלושה יתרונות של CNN על פני fully connected במשימות סיווג תמונות.
5. עבור כל אחד מסוגי data augmentation שהוזכרו ברקע התיאורטי, הסבירו מהם השיקולים לבחירה האם להשתמש בו או לא, ותנו דוגמאות מתאימות לכך.
6. ציירו באופן איכותי את אחוזי דיוק הסיווג של רשת לאורך אימון עבור סדרת האימון וסדרת הבוחן על מערכת צירים אחת, בשני המקרים הבאים:
   א. אימון ללא רגולריזציה.
   ב. אימון עם רגולריזציה (כלשהי).
   הסבירו את תשובתכם.
7. נתון מאגר תמונות בגודל 256x256 בו 49,000 תמונות צבע מ-7 מחלקות שונות. מעוניינים לסווג את התמונות למחלקות באמצעות רשת שבה יש 5 שכבות קונבולוציה בלבד ושכבת Fully Connected אחת בלבד שהיא השכבה האחרונה לפני חישוב ה-Loss (ניתן להוסיף לרשת שכבות מסוג אחר כרצונכם מלבד 6 אלו).
   א. תכננו רשת מתאימה. הגדירו את כל השכבות הנדרשות ואת הפרמטרים המתאימים להן.
   ב. מה מספר הפרמטרים הנלמדים ברשת שתכננתם?

ג. אחת מתמונות הבדיקה שהוכנסה לרשת היא בגודל 257x257x3. האם ניתן לסווג תמונה זו? במה זה תלוי? הסבירו.

8. סיווג MNIST בעזרת ReLU ו-dropouts:

א. השלימו את המימוש של שכבות ReLU ו-dropout בספריית layers-prep2 בקבצי ההכנה.

ב. היעזרו בקוד שכתבתם במעבדה הראשונה, בחלקים הראשון והשני, כדי להשלים את קבצי הרשת לסיווג MNIST המופיעים בהכנה. החליפו את שכבת הסיגמואיד בשכבת Relu שמימשתם, והוסיפו לרשת את שכבת ה-dropout שמימשתם לאחר האקטיבציה הראשונה.

ג. בסעיף זה נכבה את שכבת ה-dropout ע"י הצבת הסתברות של אפס בקובץ הראשי. הריצו את האימון כפי שהרצתם בחלק השני של המעבדה.

ד. כעת הפעילו את שכבת ה-dropout ע"י שינוי ההסתברות. בחרו את ההסתברות של ה-dropout להיות 10% והריצו את האימון שוב.

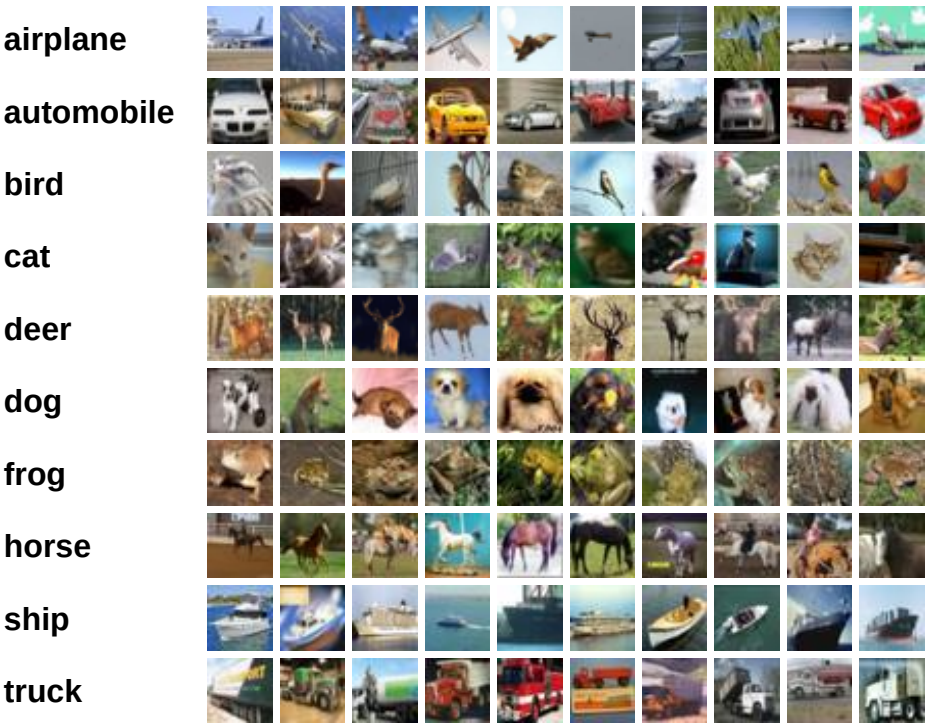
ה. רשמו את התוצאות שקיבלתם בטבלה, הוסיפו את הגרפים שמתקבלים והשוו בטבלה ובגרפים גם לתוצאת הריצה בחלק 2 במעבדה. מהן מסקנותיכם לגבי הוספת השכבות החדשות? הסבירו.

ו. הוסיפו לדו"ח כצילום מסך (ברור) את החלקים שהשלמתם מתוך הקוד עבור שאלה זו. אין צורך להגיש את הקוד עצמו.

מפגש שני – מהלך הניסוי

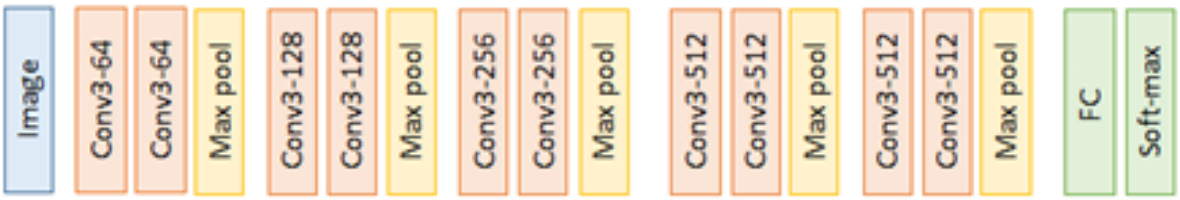
חלק א' – סיווג תמונות CIFAR10

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 נממש ארכיטקטורה הנקראת VGG13. רשתות מסוג VGG הן רשתות קונבולוציה שיכולות להיות ממומשות עם מספר שונה של שכבות ובוואריאציות שונות. באיור 21 ניתן לראות את הארכיטקטורה של רשת VGG13 לאחר התאמה עבור סיווג מאגר CIFAR10. שימו לב שבגרסה זו ישנן רק 11 שכבות נלמדות (שתי שכבות נלמדות נוספות, המופיעות בארכיטקטורה המקורית של רשת זו, אינן מופיעות כאן). למידע נוסף על משפחת ארכיטקטורות זו ניתן לקרוא כאן.



איור 21 - ארכיטקטורת VGG13 המתאימה למאגר CIFAR10.
הארכיטקטורה המלאה נמצאת באתר זה.

בשלב התאמת הפרמטרים של הרשת, מטעמי חיסכון בזמן, נסווג רק 5 מחלקות מתוך 10 המחלקות של CIFAR10. בסוף התאמת הפרמטרים נבצע אימון סופי על 10 המחלקות.

- חזרו ועברו על נספחים א׳ וב׳ בחוברת, המתארים את השכבות שבהן נשתמש בניסוי.

1. השלימו את הפונקציה VGGBlock המממשת את בלוק השכבות הראשי של הארכיטקטורה. השכבות הנחוצות להשלמת סעיף זה (וגם עבור הסעיף הבא) נמצאות בנספח א׳ ובנספח ב׳. הבלוק הראשי ישמש אותנו בבנייה של השכבות המרכזיות של הרשת בסעיף הבא.
2. השלימו את הפונקציה VGG13 המממשת את מלוא הארכיטקטורה. מהו מספר הערוצים בכל בלוק בארכיטקטורה? הסבירו באופן איכותי מה קורה לממדי המידע לאורך הרשת?
3. פתחו את הפונקציה CIFAR10Classification והתבוננו ב-10 המחלקות. בבחירת 5 מחלקות מתוכן, האם זהות המחלקות הנבחרות יכולה להשפיע על איכות הסיווג? הסבירו. לאור זאת, באלו מחלקות כדאי לבחור? בחרו 5 מחלקות מתוך ה-10 ורשמו את המחלקות שבחרתם.

לאחר השלמת מימוש הרשת ובחירת המחלקות, נבחן השפעות של היפר-פרמטרים שונים על הביצועים.

<u>שימו לב:</u> בסעיפים הבאים נפיק גרפים רבים, וכן טבלאות רבות ב-prompt של Matlab. יש להוסיף את כל הגרפים המתקבלים בהתאם לסעיפים שלהם לדו״ח המסכם, תוך ציון הפרמטרים העיקריים שאיתם הופק גרף זה. לאחר הגרף יש להוסיף את סופה של הטבלה המתאימה להרצה (כ-5 שורות אחרונות מספיקות). בנוסף לכך, יש לרשום את אחוזי הסיווג הסופיים המופיעים לאחר הטבלה.

4. תחילה הריצו את האימון ללא שינוי אפשרויות האימון ההתחלתיות.
   א. מהי קורה לגרף האימון לאחר 8 אפוקים? מדוע מתקבלת התנהגות זו?
   ב. האם הוספת רגולריזציה לרשת יכולה לעזור? נמקו.

תחילה ננסה לשפר את תהליך האופטימיזציה באימון, ע״י שינוי פרמטרי ה-scheduler.

5. שנו פרמטרים אלו בהתאם לרקע בחוברת ולהצעות המדריך. מצאו פרמטרים המשפרים את תוצאת הסעיף הקודם והישארו איתם.

כעת נבחן מספר שיטות להקטנת תופעת התאמת היתר. בשלב ראשון נבדוק את השימוש ברגולריזציית L2, בעזרת השדה L2Regularization.

6. בחרו שלושה ערכי רגולריזציה בין 0.001 לבין 0.01 ובחנו את תוצאות האימון עם ערכים אלו.
   א. לאור התוצאות שקיבלתם, מה לדעתכם תהיה התוצאה עבור ערך של 0.5? הסבירו.
   ב. מהו הערך המיטבי מבין השלושה שניסיתם? המשיכו איתו לסעיפים הבאים.

כעת נבחן את השפעת שכבת ה-dropout על האימון, בעזרת השדה dropoutProbability.

7. א. הסבירו מהי משמעות ערך ההסתברות ב-dropout. מהו ערך ההסתברות ההתחלתי?
   ב. בחרו שלושה ערכי הסתברות שונים בין 0.05 ל-0.5 ובחנו את תוצאות האימון איתם.

ג. מהי ההשפעה של שינוי ערך ההסתברות על התוצאות? הסבירו.
בחרו את ערך ההסתברות המיטבי והמשיכו איתו לסעיפים הבאים. אם לא קיבלתם שיפור לעומת תוצאת הסעיף הקודם, ניתן לבחור בערך 0.

כעת, נוסיף Data augmentation בעזרת ה-augmenter.

8. הסתכלו בקוד על השורות שמתאימות לסעיף ה-Data augmentation. ניתן להיעזר בתיעוד של MATLAB עבור האופציות המופיעות.

א. איזה סוגי Data augmentation מתאימים עבור מאגר CIFAR10? סדרו אותם לפי מידת התאמתם לדעתכם, והסבירו בקצרה.

ב. בחנו הוספת אוגמנטציות <u>באופן הדרגתי</u> לפי הסדר שרשמתם, הרקע בחוברת והצעות המדריך.

ג. איך ה-Data augmentation משפיע על התוצאות? הישארו עם השינויים שתרמו עבור הסעיפים הבאים.

כעת שנו את בחירת המחלקות בתחילת הקוד ועברו לסיווג של כל 10 המחלקות.
שימו לב שאימון יחיד של 10 מחלקות לוקח כ-6 דקות, ולכן לא מומלץ להריץ אותו מספר רב של פעמים.
ניתן להתייעץ עם המדריך לגבי ערכי ההיפר-פרמטרים שבחרתם לקראת ההרצה בשלב זה.

9. בחנו את תוצאות האימון עבור סיווג של 10 מחלקות עם ההיפר-פרמטרים שבחרתם. מה התקבל?

חלק ב' – Transfer Learning

בחלק זה נאמן רשת נוירונים לסיווג סוגי קינוחים.

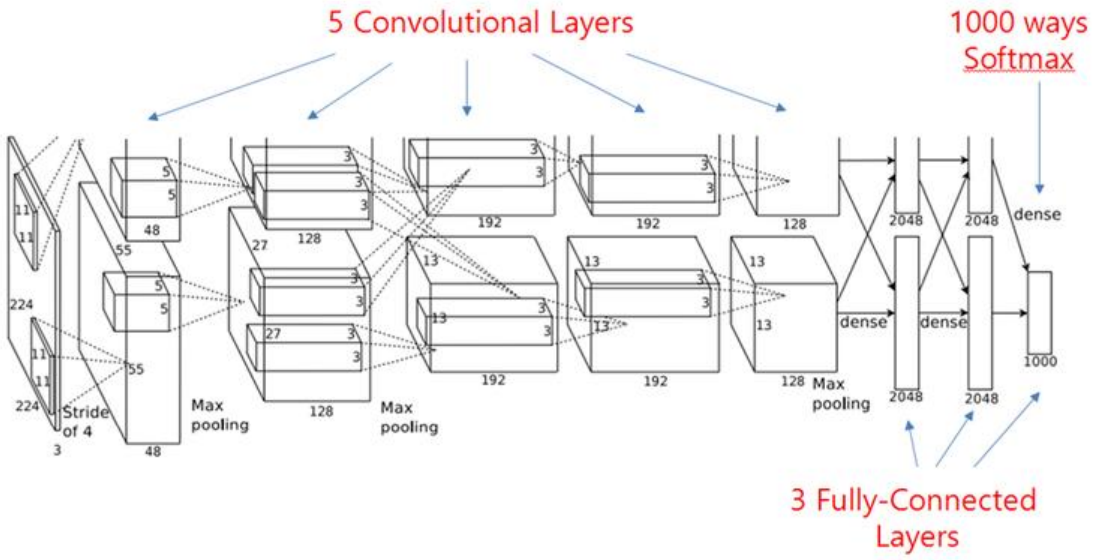
ה-Dataset למשימה זו מורכב מתמונות, ותיוג של כל תמונה אשר מעיד על המאכל בתמונה.

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).

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 ונאמנן 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.

1. מהם היתרונות של Transfer Learning? באילו סוגים של משימות ניתן להשתמש בשיטה זו?
2. מהו אופי הפלט של הריצה שאתם מצפים לקבל עם שימוש ב-Transfer Learning לעומת ריצה המאמנת רשת שלמה מנקודת מוצא נקיה?
3. השלימו את שורות הקוד במקומות המתאימים בקובץ TransferLearning.m.
   הוסיפו לדוח שלכם את פלט ההרצה והגרף המתקבל.
4. השתמשו בידע שרכשתם במעבדה זו על מנת לנסות לשפר 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.
`batchNormalizationLayer`	A batch normalization layer normalizes each input channel across a mini-batch. To speed up training of convolutional neural networks and reduce the sensitivity to network initialization, use batch normalization layers between convolutional layers and nonlinearities, such as ReLU layers.

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.