**סמינר בהנדסת תוכנה**

**אלגוריתמים בלמידה חישובית**

**רשתות נוירונים מלאכותית (Artificial neural networks)**

**אלגוריתם K מרכזים (K-means Algorithm)**

**מגישים:**

**גיל צרפתי 308214543**

**יגאל אורנס 303893416**

**מרצה:**

**ד"ר שלום מנדל**

**מבוא:**

סמינר זה עוסק בתחום הלמידה החישובית. הסמינר מחולק לשני חלקים, כאשר כל אחד חלק מוקדש לאלגוריתם יחיד אותו בחרנו מתחום הלמידה החישובית.

האלגוריתמים בהם נעסוק בסמינר:

* רשתות נוירונים מלאכותיות (Artificial Neural Networks)
* אלגוריתם K מרכזים (K-means clustering)

עבור כל אחד מהאלגוריתמים, הצגת האלגוריתם תתחלק לחלק תיאורטי ולחלק מעשי.

בחלק תיאורטי נבצע סקירה כללית של האלגוריתם, היסטוריה, ושימושים. נסביר באופן מפורט את הרעיון שעומד בבסיס האלגוריתם, השיטה, הצעדים השונים, והזרימה. בקטעים שונים לאורך ההסבר יש צורך בידע מקדים אודות מונחים מתחום עולם הלמידה החישובית. כאשר ידע כזה יידרש, נספק הסברים קצרים שיעניקו את הרקע ההכרחי להבנת האלגוריתם.

בחלק המעשי, עבור כל אחד מהאלגוריתמים נבחר בעיה או משימה שלקוחה מהעולם האמתי ונראה כיצד אנו מממשים את האלגוריתם באמצעות תוכנה (כתיבת קוד) ומשתמשים בו כדי לספק פתרון לבעיה שנבחרה.

אנו נציג את תוצאות הריצה של שני האלגוריתמים ונבחן את מידת היעילות שלו בפתרון הבעיות שהוצגו.

לסמינר זה יצורפו בנספחים מצגת הסמינר וקוד המקור של התוכניות שנכתבו.

**חלק א.1 – רשתות נוירונים מלאכותית -התאוריה**

**מבוא לנושא – מהי רשת נוירונים מלאכותיות:**

רשת נוירונים מלאכותית היא מודל מתמטי חישובי. כלומר זאת למעשה איזושהי מערכת מופשטת שמתוארת בצורה מתמטית אשר מאפשרת קבלה של קלטים שונים מצד אחד, עליהם מבצעת המערכת איזשהו תהליך חישובי והחזרה של פלט בצד שהשני, שהוא תוצאת החישוב עבור הקלטים שהתקבלו.

החידוש הרעיוני במודל של רשת נוירונים מלאכותית, ומה שמייחד אותו ממודלים אחרים, הוא העבודה שהוא פותח בהשראת המוח האנושי.

כלומר, הרעיון שעמד בבסיס פיתוח המודל היה לנסות ולחקות את האופן שבו המוח האנושי פועל. איך אנחנו חושבים? איך מתרחשת למידה? איך אנחנו פותרים בעיות?

כדי לעשות זאת, חוקרים הלכו למקור, ובחנו את האופן שבו מתרחשים תהליכים קונטיביים במח שלנו, הדרך שבה אנו לומדים, וניסו ליצור מודל מתמטי מתאים שמחקה את הפעולה.

המחקר בנושא רשתות נוירונים התחיל כבר בשנות ה- 50, כך שלא מדובר ברעיון חדש ועל אף שהניסיון למדל את פעולות מורכבת כמו חשיבה ולמידה אנושית לכדי מודל מתמטי נשמעת כמשימה קשה ביותר, המודל שהוצע היה דווקא פשוט למדי במבנה שלו. למרות שהמחקר בתחום התחיל כבר לפני למעלה מ60 שנה, התקדמות משמעותית החלה רק בשני העשורים האחרונים.

הבעיה העיקרית בתקופה שהוצג לראשונה המודל, ובעשורים שבהו לאחר מכן היא שהמחשוב לא היה חזק מספיק ולא אפשר את יישמו הלכה למעשה של המודל, כך שהוא נשאר רעיון תיאורטי בלבד. במובן זה אפשר לומר שהרעיון הקדים את זמנו. עניין זה הוביל לירידה מתמדת במחקר בתחום, והנושא כמעט נזנח לחלוטין.

משנות האלפיים, בעקבות התחזקות המחשבים ויכולות העיבוד, חלה התחדשות במחקר בתחום ורשתות הנוירונים זכו לעדנה מחודשת. המחשוב המתקדם אפשר לממש את המודל התיאורטי בדרכים שלא היו ניתנות בעבר.

העיסוק המחודש בתחום רשתות הנוירונים הוליד תת תחום חדש בתוך הלמידה החישובית הנקרא למידה עמוקה (Deep Learning) . תחום זה עוסק כל כולו במחקר של רשתות נוירונים וביישומים השונים שלהן.

רשת נוירונים כך התברר הם מודל חישובי חזק בפתרון בעיות במגוון עולמות תוכן שונים ויכול לספק אחוזי הצלחה מרשימים.

כיום רשתות נוירונים משמשות לפתרון מגוון רחב של בעיות בתחום הבינה המלאכותית בניהם בעיות מתחום זיהוי קול ותמונה ( זיהוי תווים, זיהוי כתב יד, זיהוי דיבר, זיהוי תמונה ווידאו), ניתוח והבנה של שפה טבעית, כלים לקבלת החלטות, כלים לזיהוי אנומליות בתחומים שונים (רפואה, אבטחת מידע) ועוד תחומים רבים אחרים.

בהמשך חלק זה נציג ונסביר את המודל המתמטי התאורטי, את הדרך שבה איברים ביולוגים מיוצגים כאובייקטים מתמטיים, את אופן פעולת האלגוריתם המחקה את פעולת הלמידה ולבסוף ניישם את המודל התיאורטי בתוכנה.

**רשתות נוירונים – מהמודל הביולוגי למודל המתמטי:**

כאמור, רשתות נוירונים מלאכותיות פותחו בהשארת המוח האנושי ולכן כדי להבין את הרציונל שעומד מאחורי המודל יש לבחון בתחילה את המקור- המוח האנושי.

ראשית נציין כי התיאור שנביא כאן הוא פשטני למדי ואינו בהכרח נמצא בדיוק מוחלט למודל הביולוגי אך הוא נועד להקנות את ההבנה הכללית של התרחשות תהליכי החשיבה.

המוח האנושי מורכב למעשה משלושה חלקים: גזע המוח, המערכת הליבית וקליפת המוח (הקורטקס).

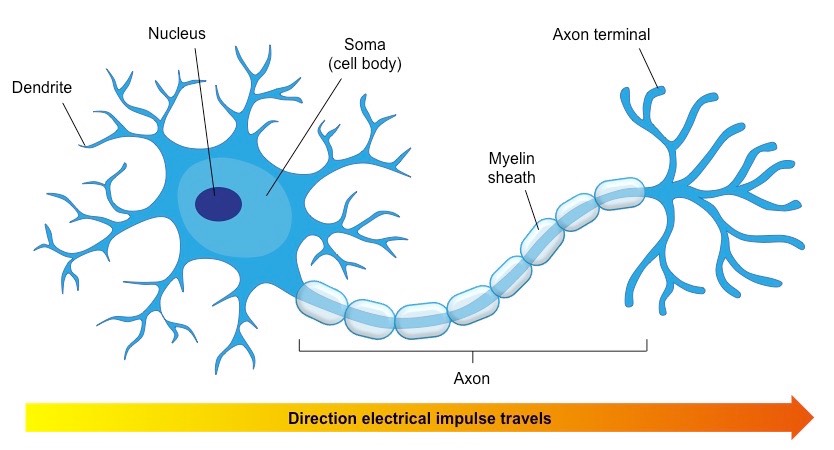
אנו נתעניין באחרון, הקורטקס. הקורטקס הוא האזור שתופס את רוב המוח האנושי ושבו נעשית החשיבה שלנו.

הוא החלק שמטפל בפעולות [קוגניטיביות](https://he.wikipedia.org/wiki/%D7%94%D7%9B%D7%A8%D7%94) גבוהות, כגון [דיבור](https://he.wikipedia.org/wiki/%D7%93%D7%99%D7%91%D7%95%D7%A8), הבנת [שפה](https://he.wikipedia.org/wiki/%D7%A9%D7%A4%D7%94) ו[קבלת החלטות](https://he.wikipedia.org/wiki/%D7%A7%D7%91%D7%9C%D7%AA_%D7%94%D7%97%D7%9C%D7%98%D7%95%D7%AA).  המקום שבו מתארגן עולם התפיסות, שבו מתוכננת פעילות, שבו נקלטת ומתפרשת משמעות הדיבור. הודות למרכיב זה של המוח אנו מסוגלים להתלבט, להשוות, להעריך ולקטלג את הרשמים המתקבלים אצלנו.

בחלק זה יש לנו כ 22 מיליארד תאי עצב, תאים אלו הם הנוירונים. כל נוירון כזה הוא למעשה יחידת עיבוד קטנטנה - יחידת התפקוד הבסיסית של המוח.

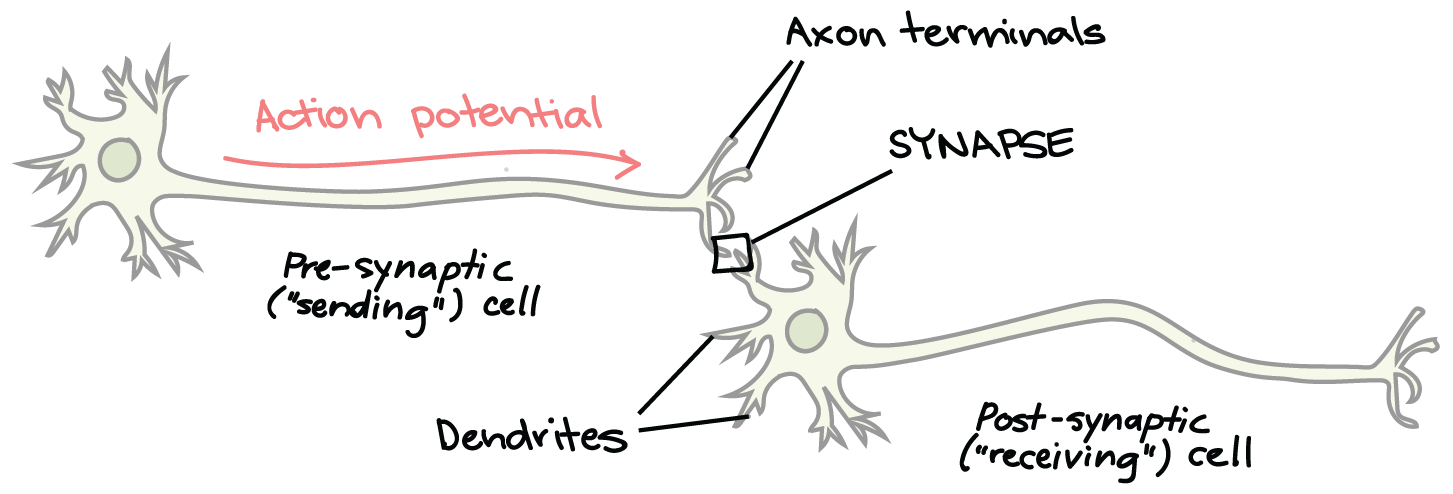
הנוירונים מקושרים זה לזה ברשת ענפה של קשרים, מיליארדי קשרים. קשרים אלה נקראים סינפסות. באמצעות הסינפסות מתקשרים הנוירונים זה עם זה ויכולים להעביר מידע מאחד לשני. הנוירונים והסינפסות יוצרים ביחד רשת חישוב סבוכה, רשת זו נקראת הרשת העצבית. זו גם הרשת אותה אנו רוצים למדל. כדי למדל אותה נבחן את מרכביה.

אם נסתכל על נוירון בודד, נראה כי מורכב מכמה חלקים:

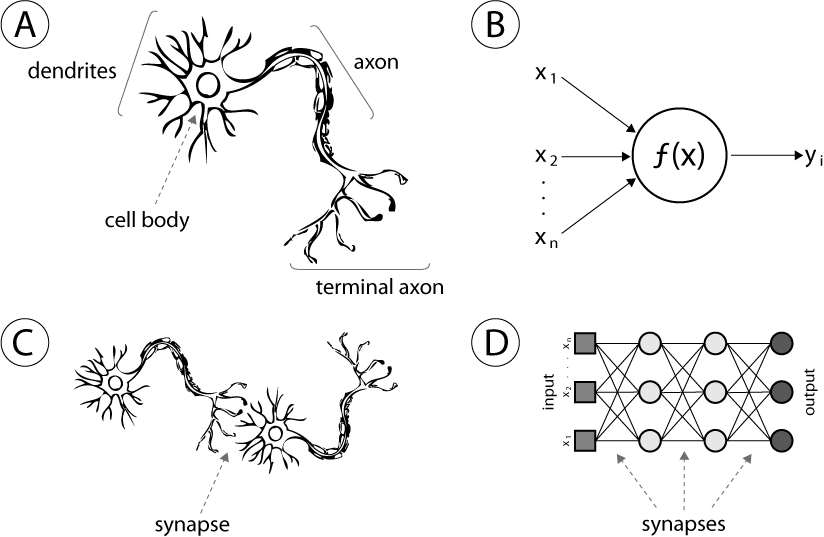


אנחנו נתמקד ב-3 חלקים עיקריים:

* דנדריט - הדנדריט הוא שלוחה קצרה של תא העצב הקולטת את האותות הנשלחים לסינפסה מתאים אחרים. הדנדריט ממיר את המידע שנקלט לאות חשמלי המכונה דחף עצבי ומוליך אותו לגוף התא. אם ננסה להקביל לרגע את הגוף האנושי למכונה, הרי שאברי החישה שלנו (עיניים , אוזניים וכו') הם החיישנים שמגיבים לגירויים חיצוניים שונים, והדנדריט הוא הרכיב שמקבל כקלט את המידע שהגיע מהחיישנים.
* גוף תא עצב – גוף תא קולט מידע חשמלי מהדנדריטים, שיכולים לשנות את הפוטנציאל החשמלי של גוף התא. גוף התא למעשה מגיב למידע שהתקבל אליו, הוא מסוגל לשלוח פרץ של אות חשמלי לעבר האקסון של התא (מתח זה נקרא [דחף עצבי](https://he.wikipedia.org/wiki/%D7%93%D7%97%D7%A3_%D7%A2%D7%A6%D7%91%D7%99) או פוטנציאל פעולה). הפרץ החשמלי הזה, הדחף, הוא למעשה פעולת העיבוד של הנוירון והתגובה שלו. הוא למעשה קובע כמה הנוירון פעיל, ותורם לפעולת העיבוד הכללית. כאשר כל נוירון הוא יחידת עיבוד קטנטנה, הכח שלהם נובע מכמות הקשרים ושיתוף הפעולה ביניהם. אפשר להסתכל על זה באופן שלכל נוירון יש תפקיד קטן, ומידת האות ששולח זו מידת ההשפעה שלו על התוצאה הכללית.
* [אקסון](https://he.wikipedia.org/wiki/%D7%90%D7%A7%D7%A1%D7%95%D7%9F_(%D7%A1%D7%99%D7%91_%D7%A2%D7%A6%D7%91%D7%99)) - האקסון הוא שלוחה ארוכה של תא העצב המתפקדת כקו תקשורת. תפקידו הוא להוליך דחפים עצביים מגוף התא הלאה אל עבר נוירונים אחרים. המידע מועבר באקסון בצורת אות חשמלי.  בקצהו של האקסון הוא מתפצל לענפים רבים, הקרויים [טרמינלים](https://he.wikipedia.org/w/index.php?title=%D7%98%D7%A8%D7%9E%D7%99%D7%A0%D7%9C_(%D7%90%D7%A7%D7%A1%D7%95%D7%9F)&action=edit&redlink=1), כאשר בקצותיהם יש כפתורים שהם נקודות החיבור לנוירונים אחרים.  אלה הסינפסות, שהן כאמור החלק המקשר בין הנוירונים.



עד כה התבוננו במודל הביולוגי. כעת נתבונן במודל המתמטי שהוצע:



**A - נוירון בודד B- נוירון מלאכותי בודד**

**C- רשת של נוירונים D - רשת נוירונים מלאכותית**

באיור הנ"ל מוצג השוואה בין המודל שהביולוגי שהוצג לבין המודל המתמטי שהוצע.

את הנוירון הביולוגי (A) מחליף נוירון מלאכותי (B)שהוא אובייקט מתמטי. לנוירון המלאכותי בדומה לנוירון הביולוגי יש כניסות, מהם הוא מקבל את הקלטים. הכניסות הם למעשה הדנדריטים בנוירון ביולוגי. את גוף התא של הנוירון הביולוגי, מחליפה פונקציית האקטיבציה, שהיא פעולת העיבוד (נרחיב בהמשך). מוצא הנוירון המלאכותי, שהיא תוצאת החישוב שלו מועברת הלאה, בהתאם פעולת האקסון של הנוירון הביולוגי. הסינפסות מיוצגות כקשרים שבין הנוירונים. קשרים אלו, נראה בהמשך, מאופיינים במשקולות, שאלו הם למעשה ערכים נומריים המבטאים את עוצמת הקשר בין הנוירונים, או את עוצמת תפקודו ותרומתו של נוירון מסוים למערכת כולה.

באיור D אנו רואים הרכבה של מספר נוירונים המקושרים זה לזה בדומה לרשת עצבית.

**רקע ללמידה חישובית מפוקחת (Supervised Learning):**

עד כה סקרנו באופן כללי מהם הן רשתות נוירונים ואת הרעיון שעומד מאחוריהן. כעת נרצה לעבור להסבר מפורט על פעולת הבניה והחישוב של המודל. כיצד מתבצע חישוב? מה הקלט ומה הפלט? איך בא לידי ביטוי תהליך הלמידה?

כדי להבין זאת יש צורך להבין מושג יסוד בתחום הלמידה החישובית שהוא למידה מפוקחת. הסיבה שצריך להכיר מושג זה היא שרשתות נוירונים כמודל חישובי, משתייך למשפחת האלגוריתמים של למידה מפוקחת.

אם ננסה להגדיר באופן כללי מהי למידה חישובית, אז נאמר שהיא היא תת תחום במדעי המחשב העוסק בפיתוח אלגוריתמים המיועדים להקנות למחשב יכולת למידה מתוך נתונים, ולבצע משימות שונות שלרוב אינן ניתנות בתכנות קלאסי.

נהוג לחלק את אלגוריתמי הלמידה החישובית למספר משפחות ,כאשר החלוקה הגסה ביותר היא לאלו המשתייכים למשפחת הלמידה מונחית(Supervised learning) ואלו המשתייכים ללמידה שאינה מונחית (Unsupervised learning).

אנו מתעניינים כרגע במשפחה הראשונה, מאחר והאלגוריתם אותו אנו סוקרים כעת משתייך אליה.

כאשר אומרים למידה מונחית או לא מונחית, הכוונה היא למעשה לסוג המידע שנותנים לאלגוריתם וכמובן לאופן שהוא בו מנצל מידע זה. באלגוריתמים השייכים לקבוצת הלמידה המונחית, אנו מספקים תמיד סט נתונים (data Set) , אשר בו, לכל דגימה, יש תווית סיווג. נסביר בדוגמה, נניח כי יש ברשותנו נתונים אודות 1000 נכסים והמחיר שבו הם נמכרו. אנו רוצים לייצר מודל שיספק לנו חיזוי, בעבור נכס שעדין לא נמכר, מה המחיר שבו הוא יימכר. במקרה הזה, סט הנתונים שסיפקו מכיל 1000 דגימות, שהן נתונים על בתים שנמכרו, ולכל דגימה כזו, יש תווית, שהיא המחיר שבו הנכס נמכר.

האלגוריתם המשתייך לסוג הלמידה המונחית חייב לקבל סט נתונים מסוג כזה על מנת לעבוד, וככל שסט הנתונים אמין יותר, כך ישתפרו הסיכויים שהמודל שנבנה יצליח לחזות בהצלחה עבור דגימות חדשות.

העיקרון של אלגוריתמים מקבוצה זו אם כך, הוא שהאלגוריתם, בדרכים שונות, לומד מתוך הסט שניתן לו, כלומר מהדוגמאות ועל פיהם יוצר איזשהו מודל. הסט הזה שמשמש את האלגוריתם ללמידה ולבניית המודל נקרא סט אימון (Training set).

לאחר שהאלגוריתם סיים ונוצר מודל חיזוי. ניתן להכניס דגימות חדשות ולקבל עבורן חיזוי.

נהוג לתת כמבחן, דגימות שאנו יודעים את התוצאות עבורן. בדוגמה שלנו, נכסים שכבר נמכרו, כדי שנוכל להשוואות בין תוצאות החיזוי כפי שהתקבלו ע"י האלגוריתם לבין תוצאות האמת. הסט שמשמש לבחינת המודל נקרא סט מבחן (Test set).

אנו נראה שהמודל של רשת נוירונים מלאכותית משתייך למשפחת האלגוריתמים של למידה מונחית. זאת אומרת, יהיה עלינו לספק מאגר נתונים ראשוני, המכיל דגימות ותוויות סיווג לכל דגימה. על סט הנתונים שניתן, תוכל הרשת העצבית המלאכותית שנבנה להתאמן וללמוד (נראה בהמשך כיצד), ובסופו של דבר לענות על שאלות שונות, נגיד בדוגמה שלנו, רשת מתאימה הייתה יכולה לענות לנו, בכמה נכס שאנו מתעניינים בו יימכר?

**רשתות נוירונים מלאכותית- ארכיטקטורה:**

כל רשת נוירונים מלאכותית מאופיינת בשלישיה הבאה:

* **חיבורים**- אופן החיבור בין הנוירונים ברשת
* **משקלים**– אלו הם ערכים נומריים המייצגים את "עוצמת הקשר". המשקלים הם לב ליבה של מערכת הלמידה. בהמשך כאשר נדבר על אופן החישוב, נבין כי זהו למעשה מידת ההשפעה של מאפיין או קבוצת מאפיינים שהם משתנים בלתי תלויים על המשתנה התלוי אותו אנו רוצים לבדוק, לחזות.
* **פונקציית האקטיבציה**- זו הפונקציה או החישוב שמתבצע בתוך הנוירון. הנוירון מקבל אוסף של נתונים/ קלטים מנוירונים אחרים, מבצע עליהם את העיבוד, כלומר את הפונקציה ומעביר את המידע הלאה.

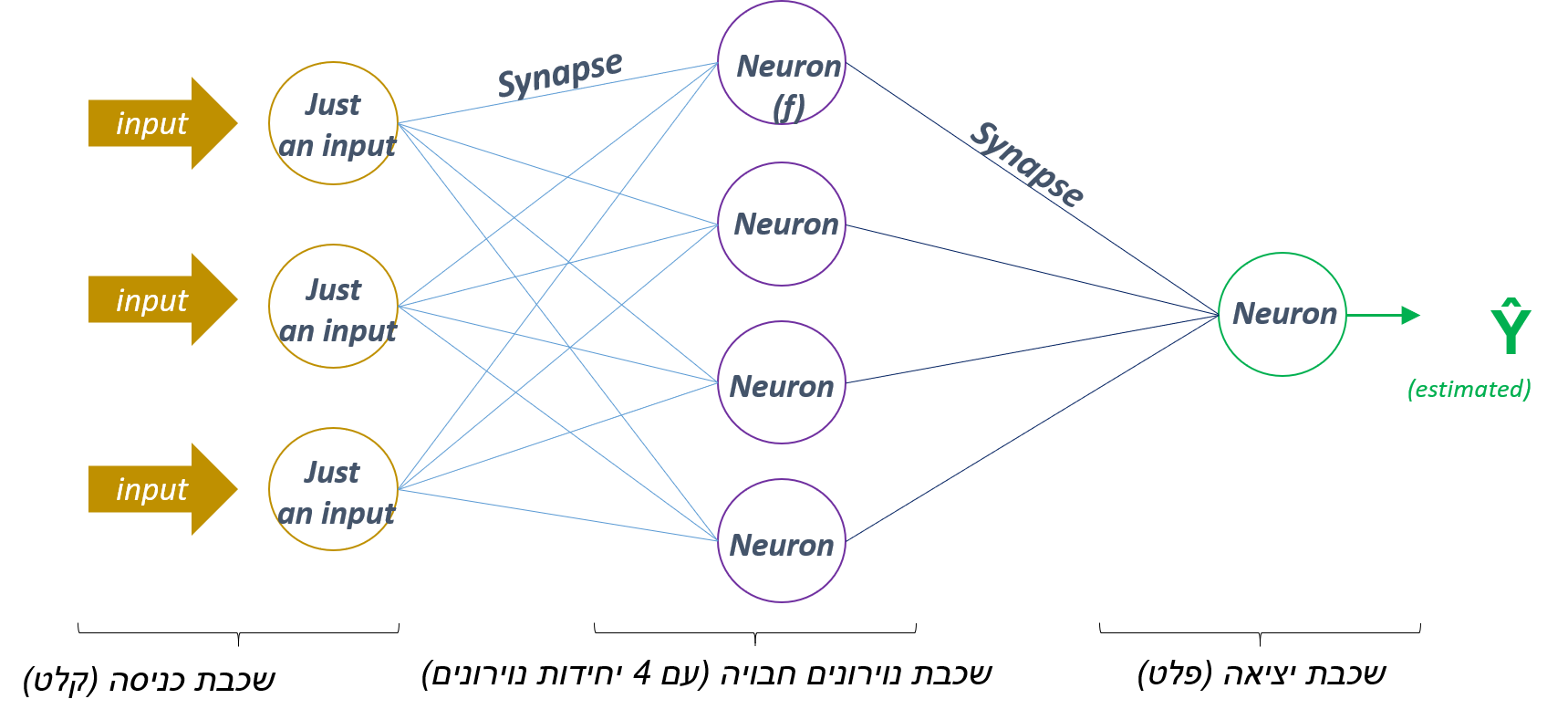
כל רשת נוירונים מחולקת לשכבות, והשכבות נחלקות ל-3 סוגים:

* **שכבת כניסה** – input layer (שכבת הקלט)
* **שכבות חבויות**hidden layers -
* **שכבת יציאה**  output layers - (שכבת הפלט)

רשתות נוירונים שונות נבדלות זו מזו, במספר הנוירונים, האופן שבו הם מסודרים (חלוקה לשכבות ומספר השכבות), במספר החיבורים, במשקלים, בפונקציות האקטיבציה ובפרמטרים שונים נוספים.

אך לכל הרשתות יש את שלושת המרכבים המצוינים למעלה, ובכולן קיימות יחידות עיבוד בדידות (המקבילה לנוירונים הביולוגיים) הקשורות ביניהן בקשרים מרובים.

כעת נסתכל על רשת נוירונים לדוגמה:



בדוגמה הנתונה אנו רואים רשת נוירונים פשוטה, עם שכבת כניסה בעלת 3 נוירונים, שכבה חבויה אחת בעלת 3 נוירונים ושכבת פלט עם נוירון בודד.

אנו נספק לרשת סט אימון שעליו תוכל הרשת להתאמן וליצור את המודל שלה.

לשכבת הקלט, נכנסת בכל פעם דגימה בודדת. כמות הנוירונים בשכבת הכניסה היא בהתאם לכמות המאפיינים/ תכונות בדגימה שמתקבלת.

נניח בעבור הדוגמה שהצגנו קודם, אם דגימה מכילה פרטים אודות נכס, והתווית את המחיר שבו הנכס נמכר, אזי אם הרשת הנתונה מעלה מתאימה לסט הנתונים של הנכסים, ונניח כי לנכס יש מידע על שלושה מאפיינים שלו. נניח, גודל הנכס, מיקום הנכס, וגיל הנכס.

כל אחד מהנתונים הללו יכנס לנוירון משלו.

במוצא הרשת, בשכבת היציאה, ישנו נוירון יחיד במקרה של הדוגמה (זה כמובן לא חייב לא כך וזה תלוי בשאלה שבודקים). במקרה שלנו נקבל במוצא איזשהו ערך שהוא חיזוי למחיר המכירה הצפוי של הנכס.

אם זהו נכס שכבר נמכר, כמו בנתונים של סט האימון, נוכל להשוואות את התוצאה למחיר המכירה האמתי ולקבל מידע אודות מידת הדיוק/ הטעות בחיזוי. בהתאם לכך נשאף ליצור תהליך שיוכל לשפר את המודל בהתאם לשגיאה שהתקבלה.

**הנוירון ופונקציית האקטיבציה:**

כעת נתמקד בנוירון יחיד ונראה כיצד הוא נראה. באיור הבא אנו רואים המחשה לנוירון מלאכותי. הנוירון מקבל כקלט את הפלטים של כל הנוירונים אליהם הוא מחובר משכבה הקודמת לו (או קלט חיצוני אם מדובר בשכבת הכניסה).

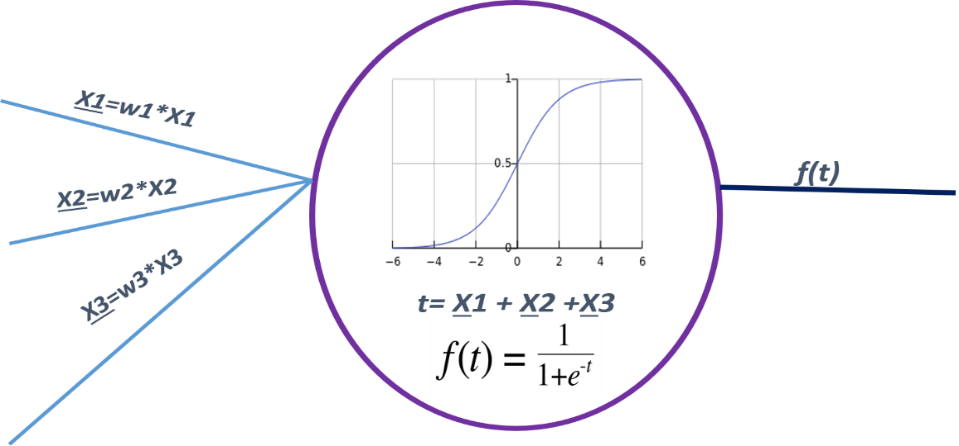
**הנוירון מבצע שתי פעולות:**

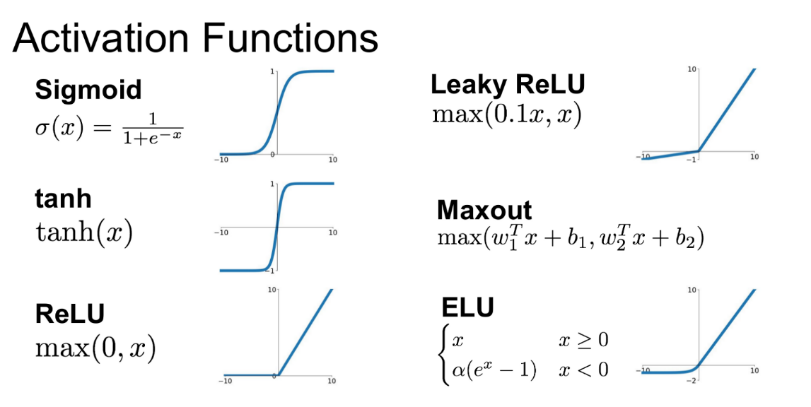
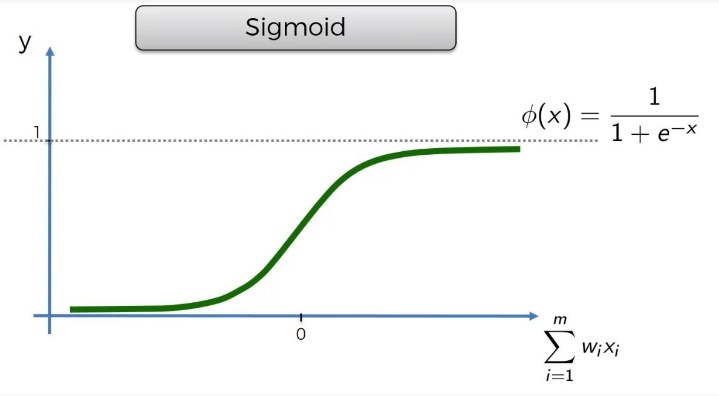
האחת היא סכימה משוקללת של סך כל הקלטים שנכנסו אליו. כלומר כל קלט מוכפל במשקולת שנמצאת על הקשר שמחבר אותו לנוירון.

הפעולה השנייה היא הפעלת פונקציית האקטיבציה.

מסיבות מתמטיות, נהוג בדרך כלל להשתמש בפונקציה המניבה ערכים בין 0 ל1 או בן 1 ל 1-.

בדוגמה שבאיור מוצגת פונקציית הסיגמואיד שמשמשת באלגוריתמים רבים בתחום הלמידה החישובית. היתרון בפונקציה זו שהיא נותנת ערכים רציפים בין 0 ל1 שיכולים להתפרש כהסתברות לתוצאה מסוימת. כך אנחנו יכול לתת לחיזוי שלנו, גם אחוז שמייצג את מידת הוודאות של המערכת בחיזוי שניתן. יש לציין שיש פונקציות נוספות שיכולות לשמש כפונקציית אקטיבציה, כאשר ההחלטה היא בהתאם לעולם הבעיה, מבנה הרשת, המידע המבוקש (הסתברותי, בינארי וכו') וגם על ניסוי וטעייה.





**איך רשתות נוירונים מחשבות:**

החישוב בפועל של רשתות נוירונים מתבסס בעיקר על אלגברה ליניארית וכפל מטריצות. נתבונן בדוגמה ונסביר כעת כיצד מתבצע החישוב.

המטריצה X היא מטריצת הקלט שלנו. היא סט האימון של המערכת. במטריצה הזו יש 7 שורות, שהן 7 דגימות. לכל דגימה יש 3 מאפיינים שמייצגים אותה.

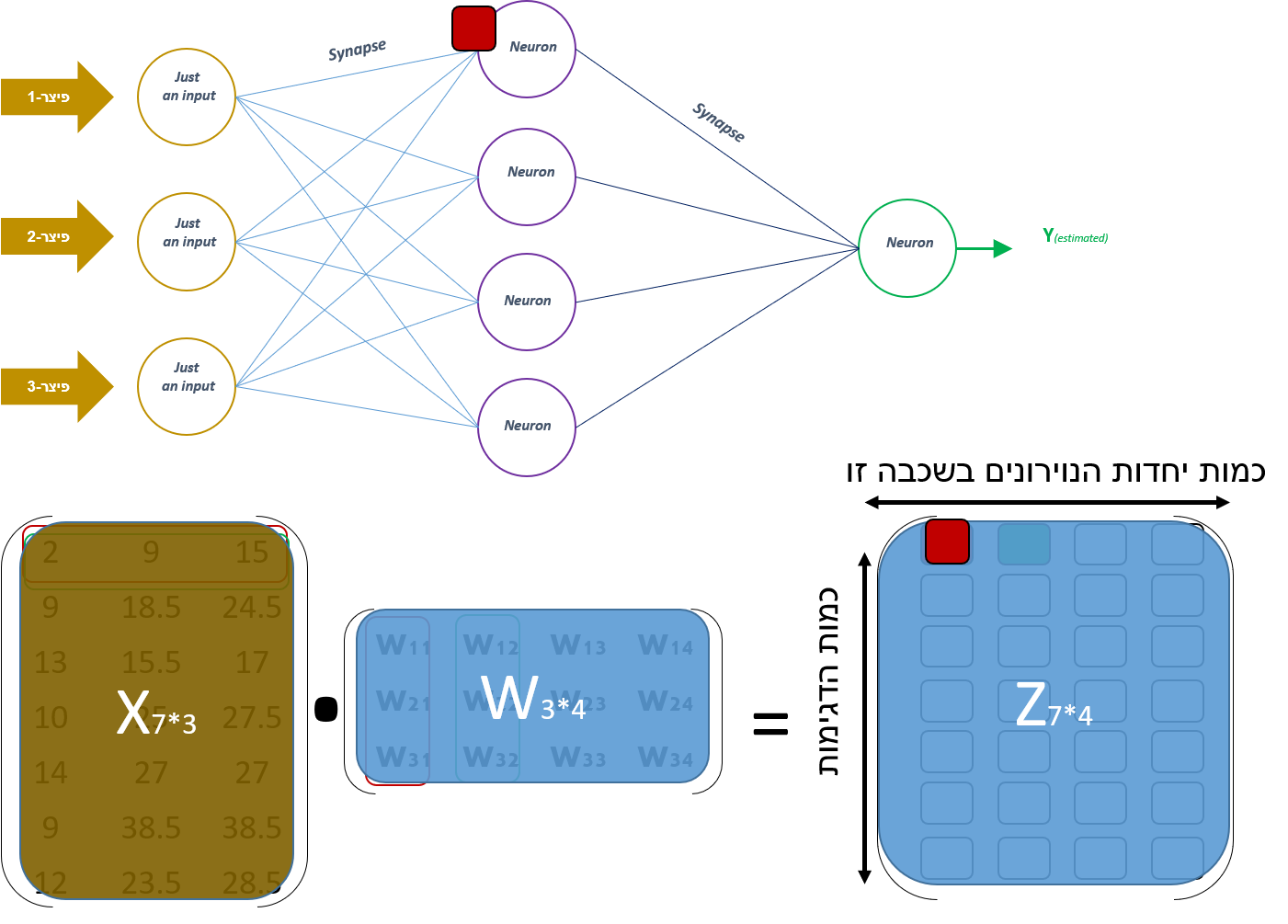
המטריצה W מייצגת את הקשרים, או הסינפסות. W12 לדוגמה מייצג את הקשר שבין נוירון מספר1 בשכבת הכניסה לנוירון מספר 2 בשכבה החבויה.

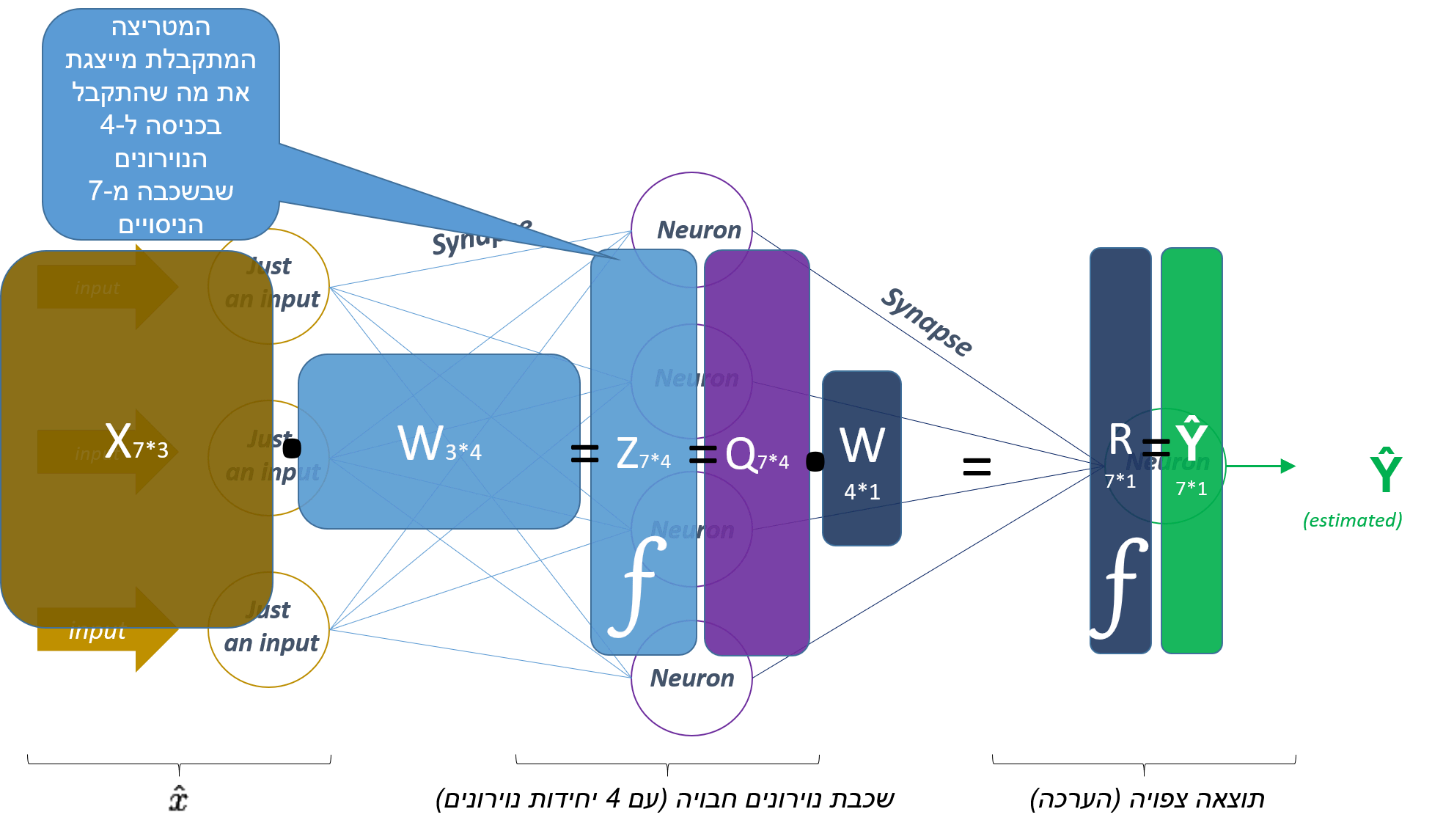
המטריצה Z היא מטריצת התוצאה של הכפלת שכבת הכניסה במשקולות.

מטריצה זו מייצגת את התוצאות עבור כל הנוירונים בעבור כל אחד מהדגימות.

החלק הצבוע באדום מייצג את התוצאה שהתקלה לנוירון הראשון בשכבה החבויה מהדגימה הראשונה.

באופן כללי, אם נסתכל על העמודה במטריצה Z, היא מכילה את כל התוצאות שהתקבלו לנוירון ה-i עבור כל אחת מהדגימות.

באופן דומה, אם נסתכל על השורה הj היא מכילה את כל התוצאות לכל הנוירונים בשכבה עבור תצפית בודדת, תצפית הjי-ת.



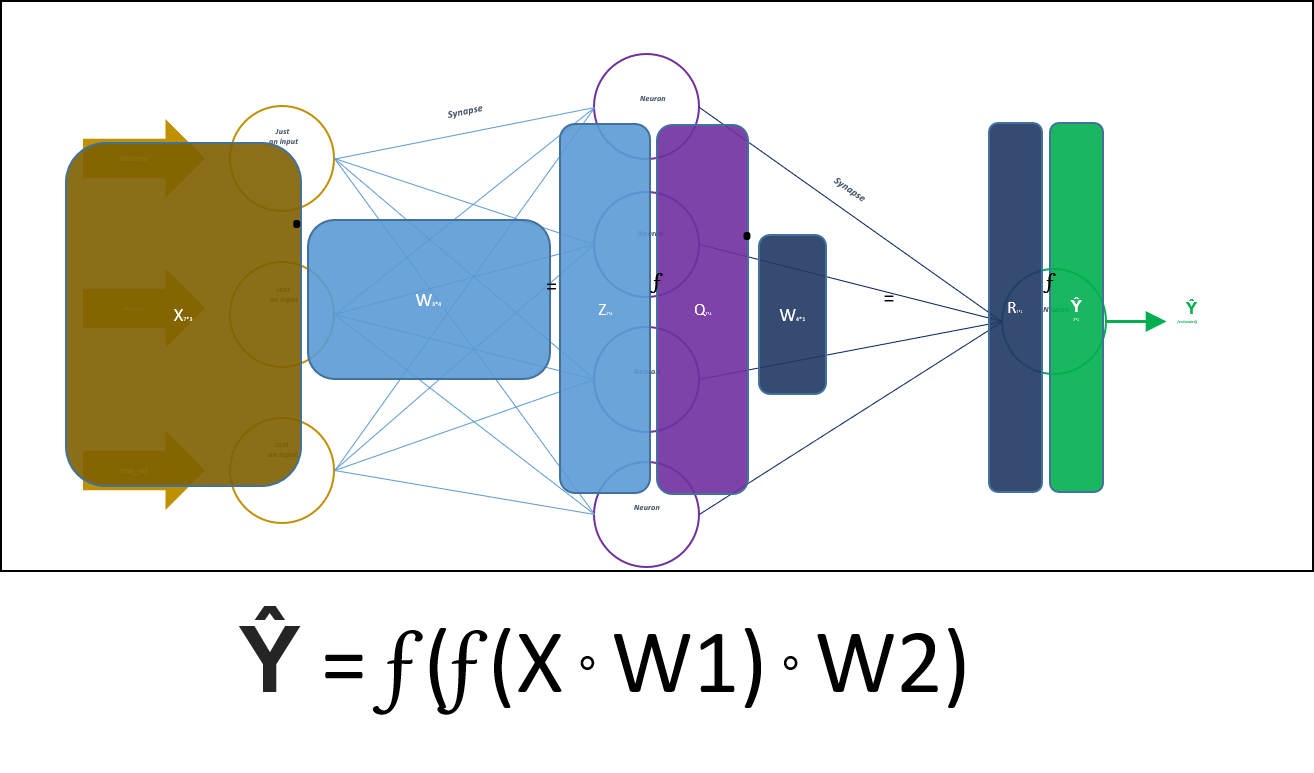
כאשר המטריצה Z, שמייצגת כאמור את הקלט לשכבת הנוירונים השנייה (השכבה החבויה), מגיעה לשכבת הנוירונים, תהליך זהה לתהליך שתואר מתרחש עבור השכבה הבאה.

הנתונים עוברים בפונקציית האקטיבציה ו מתקבלת המטריצה Q שהיא הפלט של השכבה החבויה. הנתונים בה מוכפלים במטריצת המשקולות ומקבלים את הווקטור R.

פונקציית האקטיבציה מופעלת שוב ונקבל את הווקטור Y.

הווקטור Y הוא הפלט של המערכת והוא מייצג את החיזויים עבור כל אחת מן התצפיות.

מאחר והיו 7 תצפיות במטריצת הקלט, נקבל וקטור של 7 חיזויים. כאשר כל חיזוי הוא בהתאמה לתצפית.



איור זה מסכם את התהליך של מעבר בודד של כל סט הנתונים ברשת. כל שכבה מוכפלת במטריצת המשקולות ומבצעת את פונקציית האקטיבציה על התוצאה. התוצאה של שכבה אחת היא הקלט של השכבה הבאה. כך עד שמגיעים לשכבה האחרונה שתוצאתה הוא וקטור החיזוי

**איך רשתות נוירונים מתאמנות)לומדות):**

עד כה ראינו כיצד מתבצע חיזוי עבור סט הנתונים הנתון. אך לא הזכרנו כיצד נבחרות המשקלות? מה קורה עם החיזוי אינו מדויק? איפה בא לידי ביטוי הלמידה. הכח במודל טמון במשקולות. תהליך הלמידה של הרשת הוא התהליך של עדכון המשקולות.

למעשה המשקולות, הן הדבר היחיד במודל, שניתן להשפיע עליו לאחר שמבנה הרשת נקבע, כלומר לאר שנקבעה טופולוגית השכבות, פונקציית האקטיבציה וכו'.

לפני שנראה את האופן המתמטי של תהליך עדכון המשקולות, ננסה לתאר את הרעיון באופן כללי.

כאשר אנו בונים את הרשת, עוד לפני שהרצנו עליה את הנתונים, אנו קובעים את ערכי המשקולות(סינפסות) לערכים רנדומליים.

כעת נזרים את סט הנתונים בתוך הרשת בהתאם לתהליך שתואר קודם, אשר בסופו נקבל וקטור של ערכים חזויים.

עבור כל אחד מהערכי החיזוי שנקבל נבדוק אותו מול ערך האמת ונמדוד את השגיאה.

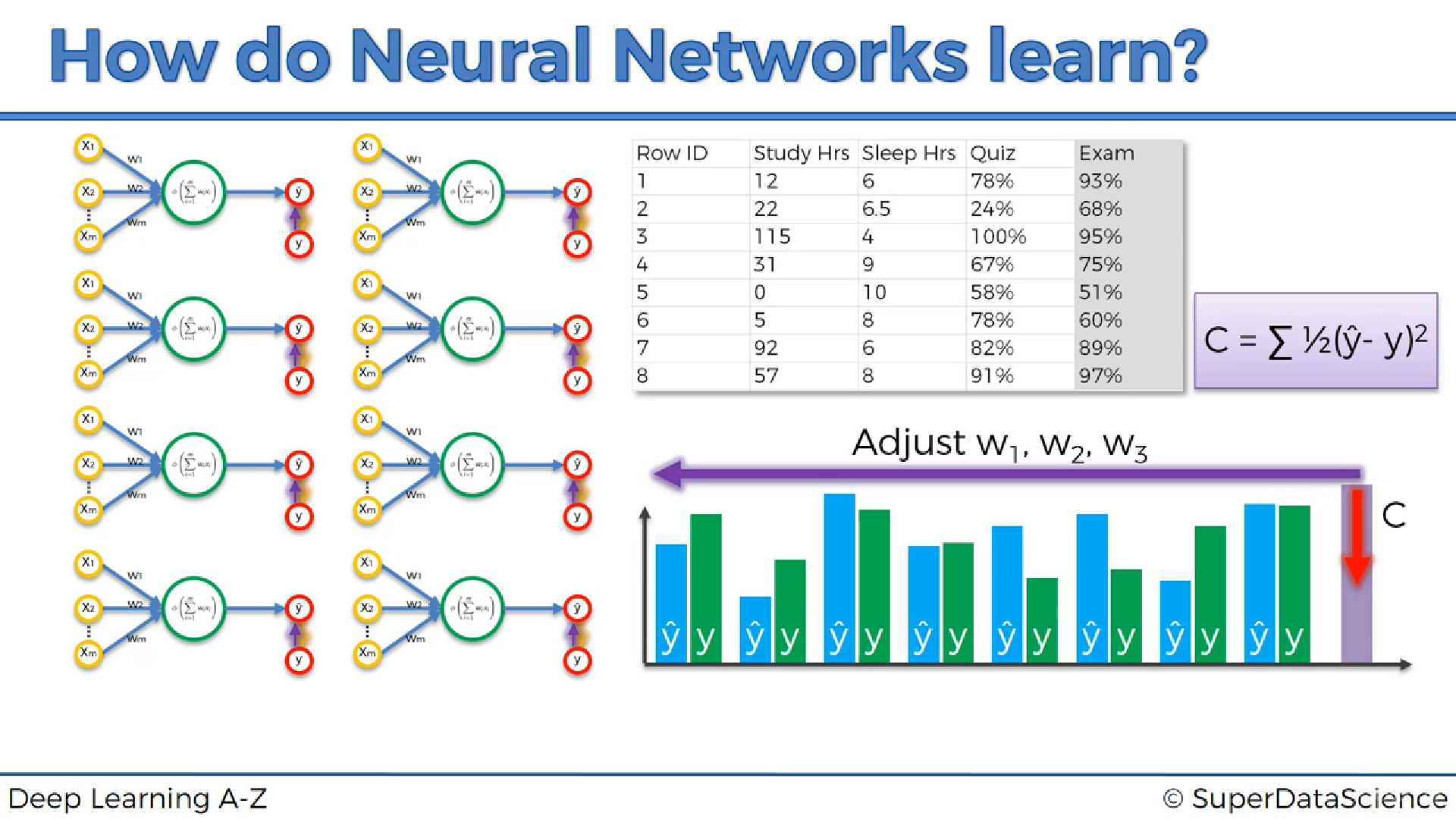
אם אין שגיאה, זה אומר שהמודל שלא לא טעה בחיזוי, אך מקרה כזה לא סביר שיתרחש עבור הערכים הרנדומליים של המשקלות שהצבנו.

סביר יותר שתהיה לנו טעות. את הטעיות הנ"ל נסכום ונקבל מדד לשגיאה.

אנחנו נרצה להקטין את השגיאה ככל הניתן. כלומר נשאף שהמודל ייתן תחזיות שהן זהות לתוצאות האמת.

מאחר ואנחנו יודעים את תוצאות האמת, נעבור שוב על הרשת, הפעם בכיוון ההפוך, כלומר נתחיל בשכבת הפלט שעבורה אנו יודעים את התוצאה הרצויה (תוצאת האמת של התצפית) ונעדכן את המשקולות אחורה עד שנגיע חזרה לשכיבת הכניסה. תהליך זה שבו הנתונים עוברים מצד אחד לצד שני של הרשת ולאחר מכן מעדכנים את המשקולות בכיוון ההפוך, זהו תהליך הלמידה של הרשת.

המעבר בכיוון הרשת נקרא Forward Propagation. המעבר בכיוון ההפוך על מנת לעדכן את המשקולות נקרא .Backpropagation בדרך כלל אנו נריץ תהליך זה באופן אינטרטיבי. כלומר כל סט הנתונים יעבור מספר פעמים ברשת, כאשר בסיום כל מעבר של כל הנתונים ברשת אנחנו נעדכן את המשקולות.



בתמונה הנ"ל מתוארת איטרציה בודדת של תהליך האימון. בטבלה הנתונה מתארת סט אימון הכולל 8 תצפיות אודות נתונים על סטודנטים. לכל תצפית שני 3 מאפיינים: מספר שעות הלימוד, מספר שעות השינה וציון בבוחן. לכל דגימה יש גם תווית סיווג, שהוא המשתנה אותו אנו רוצים לחזות והוא גם תוצאת האמת של הדגימה. בדוגמה הזאת מדובר בציון שיקבל הסטודנט במבחן.

הרשת שמתוארת היא רשת פשוטה ביותר (אפשר לומר שהיא אינה ממש רשת מאחר ואין לה שכבה חבויה). רשת פשוטה כזו נקראת פרספטרון (Perceptron), והדוגמה הזו ניתנה רק לשם פישוט תהליך ההסבר.

שכבת הכניסה של הרשת מכילה שלושה נוירונים, בהתאמה למספר המאפיינים של כל דגימה בסט האימון. באיור הרשת מופיעה שמונה פעמים, אך זאת רק לצורך המחשה. יש רק רשת אחת ובכל פעם דגימה בודדת (שורה בטבלה) זורמת ברשת.

עבור כל שורה, נקבל ערך חיזוי ונחשב את השגיאה ע"י פונקציית שגיאה הריבועית.

לאחר ששכל הדגימות יעברו ברשת. יהיה בידנו את השגיאה המצטברת שמופיעה באיור בסימון C.

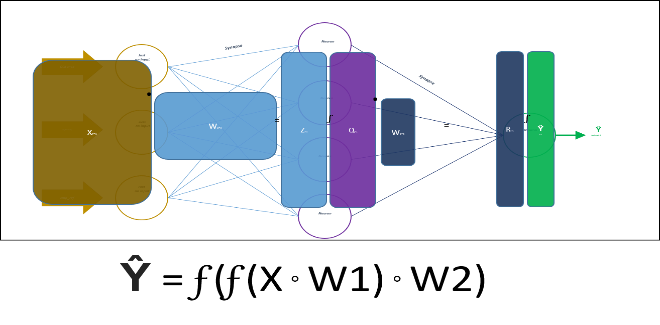
המטרה שלנו כמובן היא להוריד את השגיאה C למינימום, ולכן נעדכן את המשקולות.

לאחר שעדכנו את המשקולות סיימנו איטרציה בודדת. איטרציה כזו, שבה כל סט הנתונים עבר ברשת מכונה epoch.

**כיצד אנו מקטינים את השגיאה- שיטת Gradient Descent:**

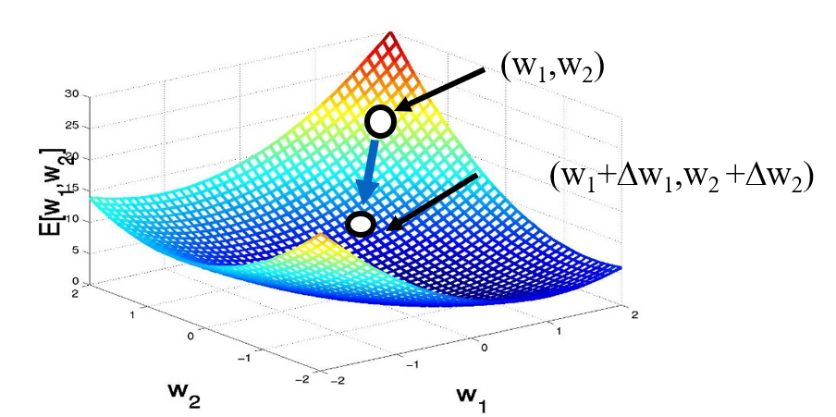
כדי להקטין את השגיאה, עלינו כאמור להתאים את המשקולות. השאלה הנשאלת היא, בהינתן פונקציית השגיאה, תוצאת החיזוי, ותוצאת האמת, איך אנו מחליטים אילו ערכים חדשים לתת למשקולות?

איך נדע אילו ערכים באמת יקטינו לנו את השגיאה, ולא נניח דווקא יגדילו אותה.



איך הטעות תשתנה אם נשנה את ערך המשקולות?

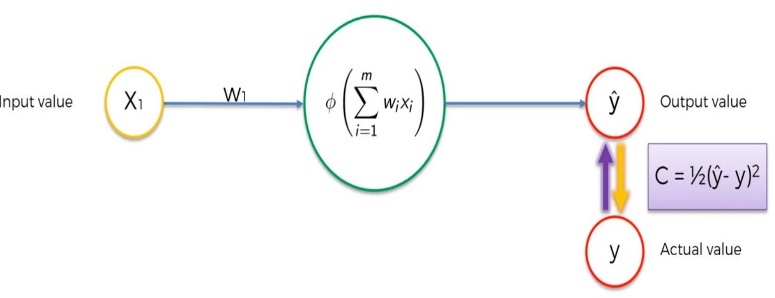
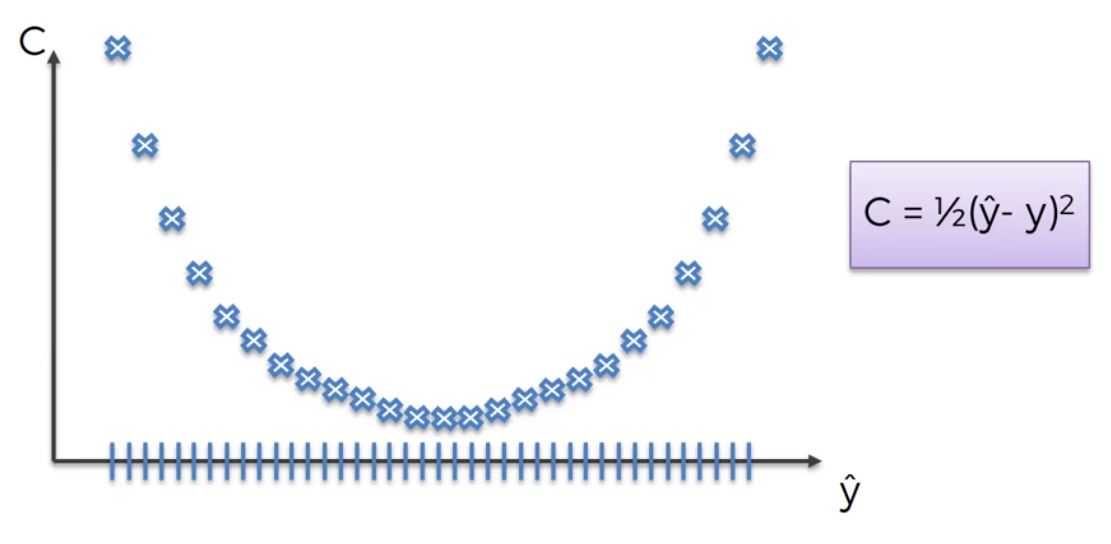
מרחב הטעות מיוצג גרפית כתלות בערכי המשקולות.

באיור מסומן המיקום הנוכחי עבור ערכי המשקולות הקיימים

ונקודות המינימום אליה אנו שואפים להגיע.

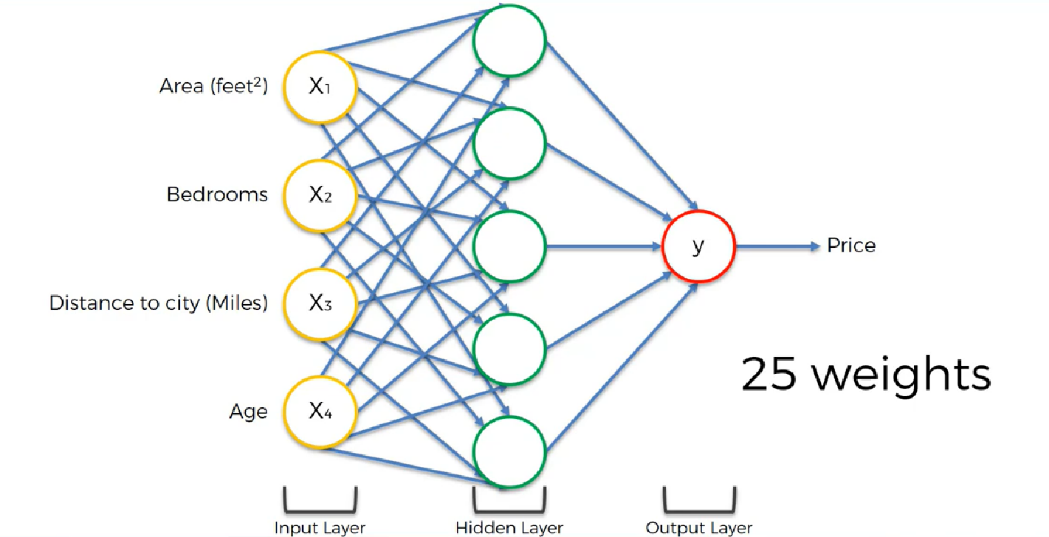
בגישה נאיבית היינו יכולים לנסות הרבה ערכים שונים של משקולות, ולבדוק איזה קונפיגורציה של משקולות נתנה את התוצאה הטובה ביותר, הווה אומר את השגיאה הקטנה ביותר והחיזוי המדויק ביותר.

נסתכל נניח על בדוגמה הפשוטה ביותר שיכולה להיות. באיור הבא נתון פרספטרון (רשת ללא שכבה חבויה) והגרף שמתאר את השגיאה כתלות בתוצאת החיזוי:



תיאורטית היינו יכולים לנסות מספר גדול של ערכים שונים לW1, נניח 1000 ערכים שונים (למרות שבפועל יש אינסוף ערכים שניתן לבדוק) ולבדוק איזה ערך נתן לנו את התוצאה הטובה ביותר. אך מה אם היה לנו רשת מורכבת יותר, ומספר המשקולות היה גדול יותר.

נסתכל על הדוגמה הבאה:



ברשת הנתונה יש 25 קשרים(משקולות). גם זו אינה רשת גדולה יחסית למה שמשתמשים ביישומים פרקטיים בתעשייה, שם מספר הקשרים יכול להגיע למאות ואלפי קשרים.

נניח ונרצה גם כאן לבדוק 1000 ערכים שונים לכל משקולות. אנחנו נגלה שבגלל כמות המאפיינים מספר הקומבינציות האפשריות גדול בסדרי גודל. בחישוב פשוט יש לנו 1000 אפשריות לכל משקולות, ו25 משקולות, כאשר כל משקולות בלתי תלויה באחרת, ונקבל שיש לנו בסה"כ 1000 בחזקת 25 קומבינציות שונות, שהם 10 בחזקת 75 קומבינציות.

חישוב בסדר גודל כזה, גם עבור המחשוב החזק ביותר שקיים היום, אינו ישים (ייקחו מיליוני שנים בתאוריה).

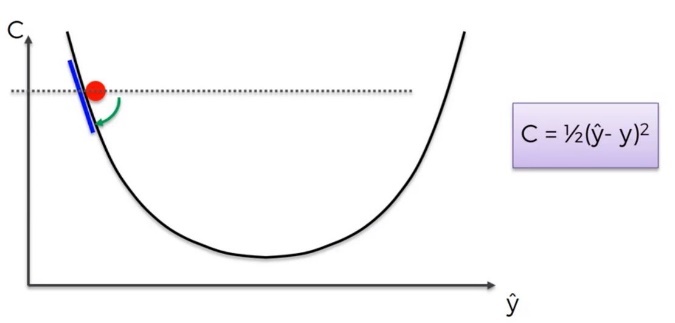
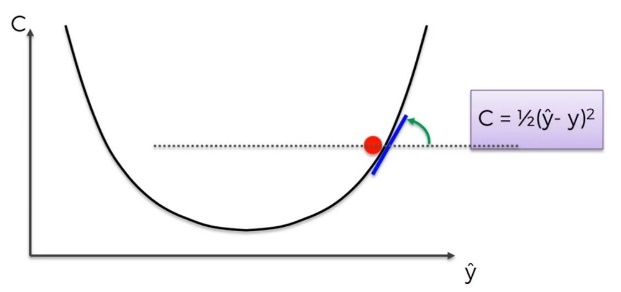
אם כך, נדרשת גישה לטכניקת קביעת הערכים החדשים למשקולות. טכניקה זו נקראת שיטת **Gradient Descent**.

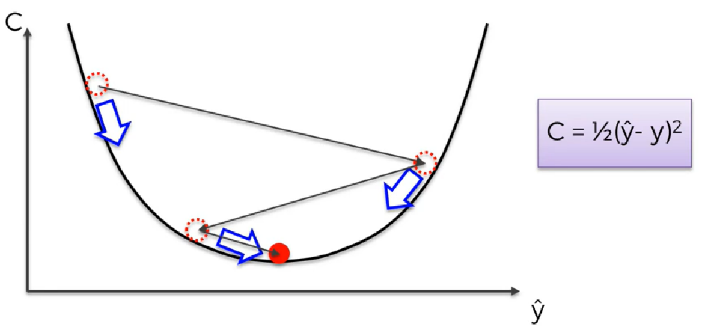
שיטת ה Gradient Descentמבוססת על אופרטור הגרדיאנט. גרדיאנט הוא הכללה של המושג נגזרת בעבור פונקציות המכילות מספר משתנים. תוצאת הגרדיאנט היא וקטור כלומר יש לו כיוון.

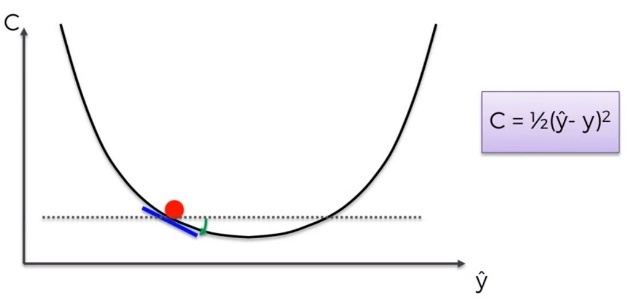
הכיוון שאליו ווקטור הגרדיאנט מצביע הוא הכיוון בו השינוי בשדה (פונקציה) עליו הוא פועל הוא מקסימלי.

שיטה ה- Gradient Descent מנצלת תוכנה זו של וקטור הגרדינאט, כדי לדעת את הכיוון שבו יש לעדכן את המשקולות. בשיטה, נעשה צעד נגדי לגרדיאנט ביחס לנקודה הנוכחית.

נסתכל בדוגמה ויזואלית כדי להמחיש זאת:







בכל נקודה נחשב את הגרדיאנט, מאחר ומדובר בפונקציה עם משתנה יחיד, במקרה הזה הגרדיאנט הוא פשוט הנגזרת. עבור חישוב זה נקבל את השיפוע או השתנות הפונקציה בנקודה. מידע זה יאפשר לנו לדעת האם עלינו לצעוד שמאלה או ימינה כדי להתקרב למינימום. בעצם הגרדיאנט מגדיר לנו איך עלינו לעדכן את המשקולות.

באיור האחרון ניתן לראות את צורת הזיגזג שנוצרה מהתהליך. גודל הצעד וכתוצאה מכך מספר הצעדים שאנו עושים נקרא קצב הלמידה (learning rate). באופן כללי אפשר לומר שאם נבחר קצב למידה איטי, ייקח לרשת צדעי חישוב רבים יותר ולכן זמן רב יותר עד שנתכנס למינימום. מצד שני, אם נבחר קצב למידה מהיר מידי, יתכן שבמקום להתכנס למינימום, נקבל התנהגות מתבדרת או שניקלע למינימום מקומי (תלוי במרחב השגיאה).

להלן נוסחת הצעד הבא שיטה ה- Gradient Descent:

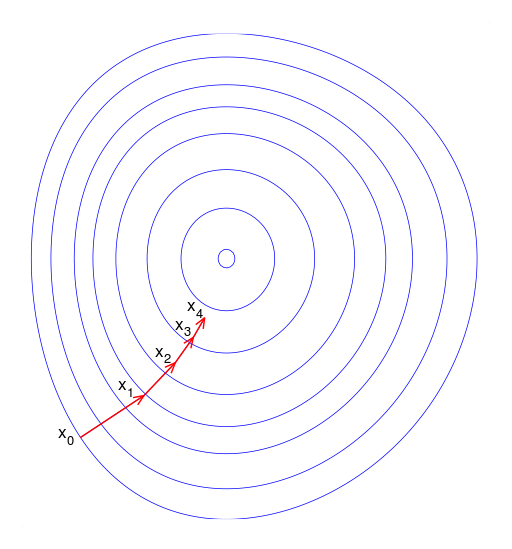


נשים לב שהצעד הבא מחושב ביחס לנקודה הנוכחית. הצעד נעשה בכיוון נגדי לכיוון הגרדיאנט. במילים אחרות, הביטוי מוחסר מa- מכיוון שרוצים לזוז נגד כיוון הגרדיאנט, מטה לכיוון המינימום.

מסמן את גדול הצעד שאחראי כאמור על קצב הלמידה.



הדוגמה בתמונה מתארת פונקציה שגיאה פשוטה, כמובן שניתן להכליל זאת למספר רב של ממדים, אך מעבר לשלושה ממדים לא נוכל להציג זאת בצורה ויזואלית.



הצגה ויזואלית של אופטימיזציה באמצעות שיטת Gradient Descent

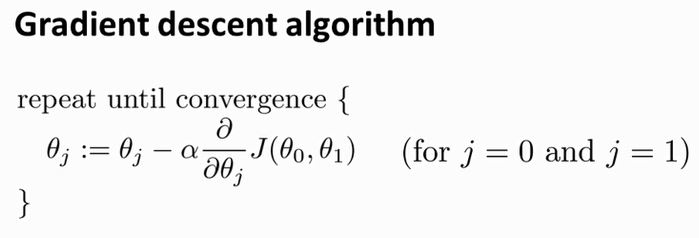
במרחב עם שני משתנים (הדוגמה הקודמת הייתה במישור,

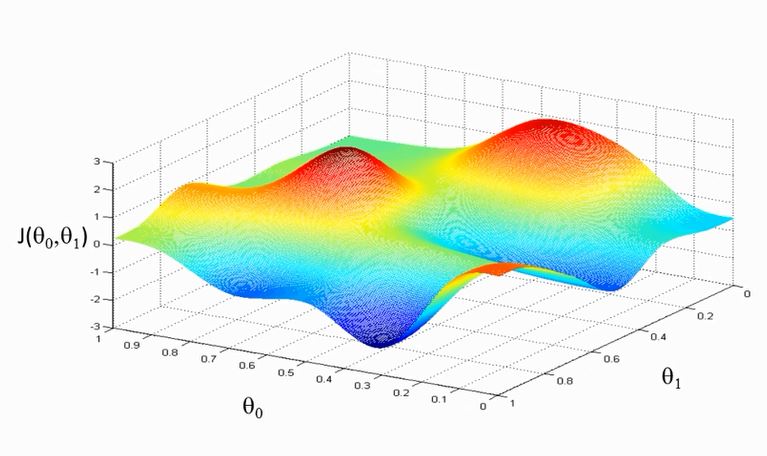
כלומר משתנה בו דד). הקווים הכחולים מתארים

את עקומת קווי הגובה. האיקסים הם סדרת

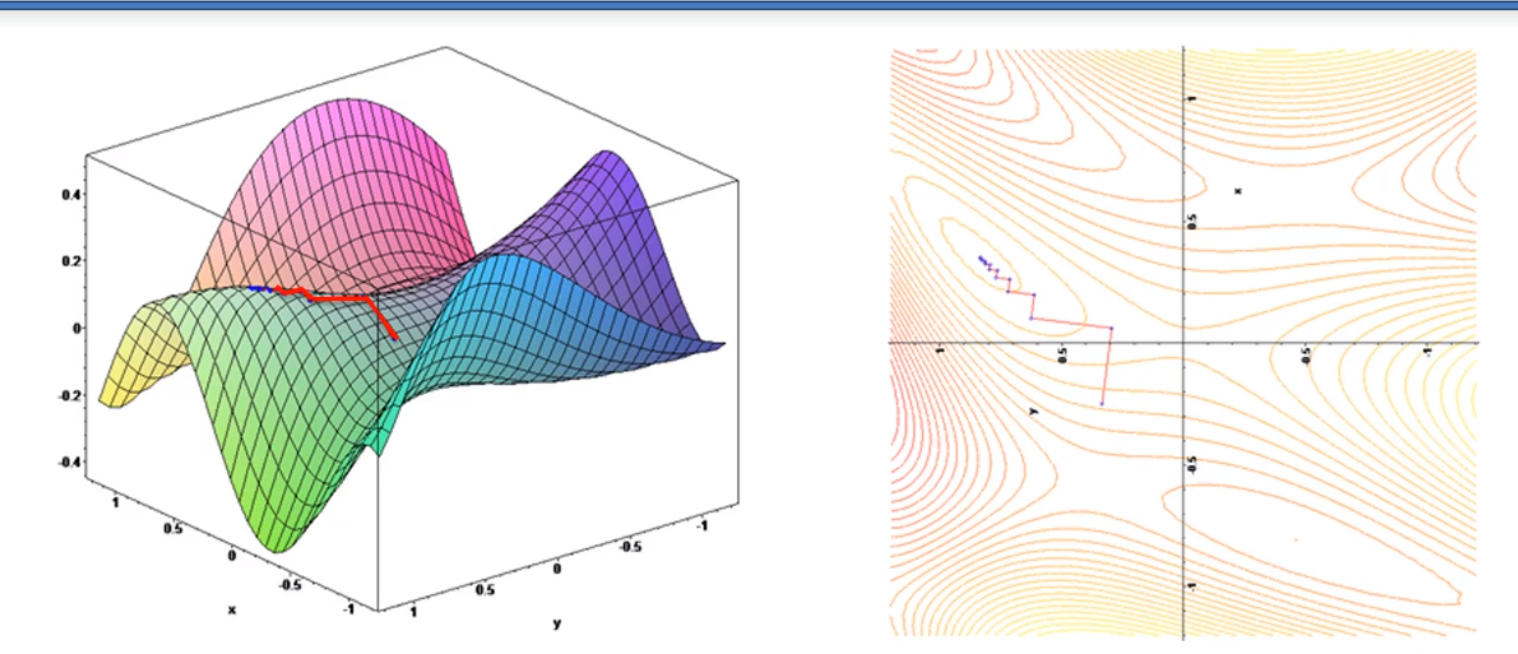
הנקודות שהתקדמנו, כאשר החץ האדום מסמל את כיוון

ההתקדמות, שהוא כנגד כיוון הגרדיאנט באותה נקודה.





שיטת הגרדיאנט במרחב תלת מידי:

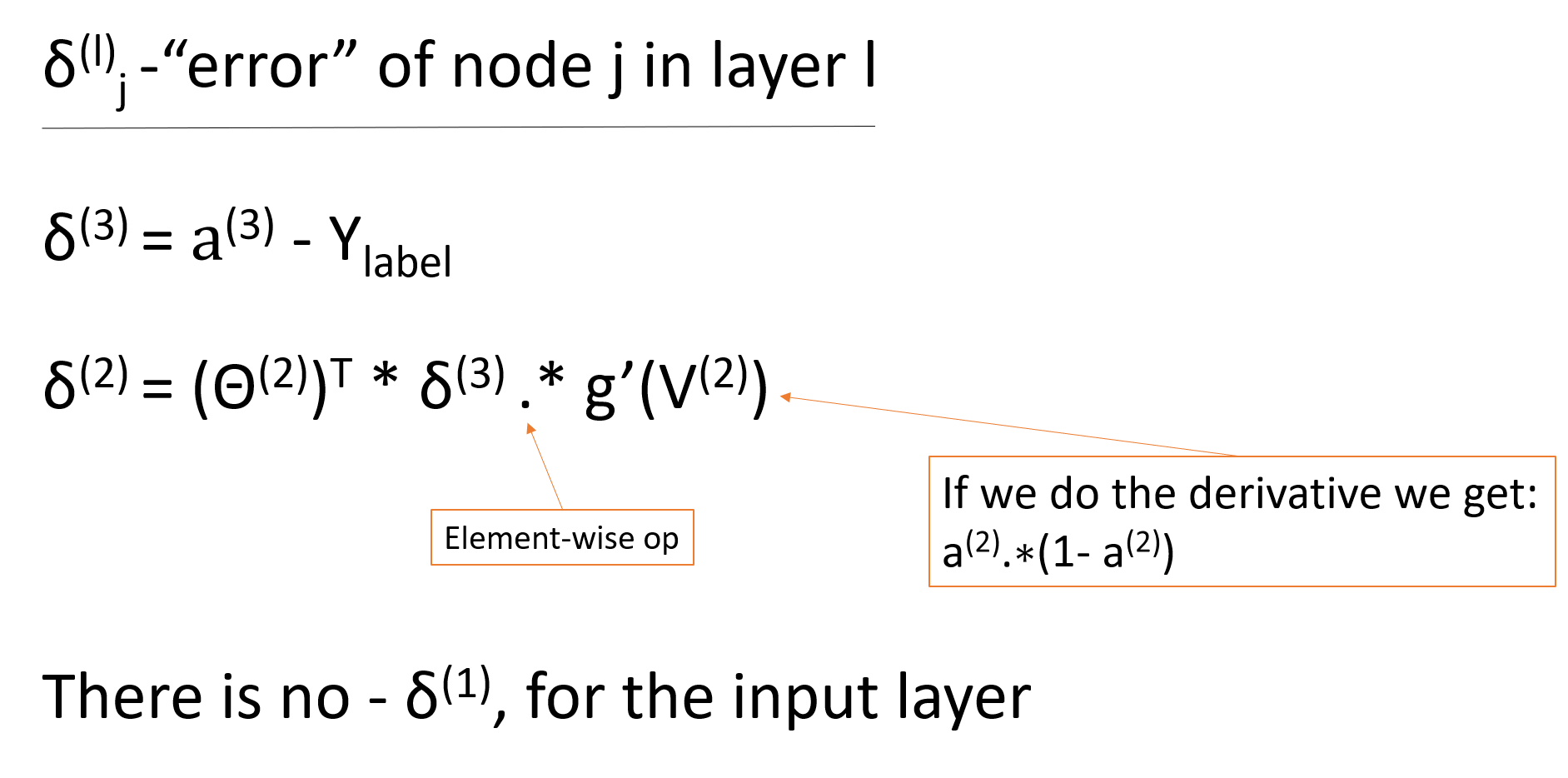


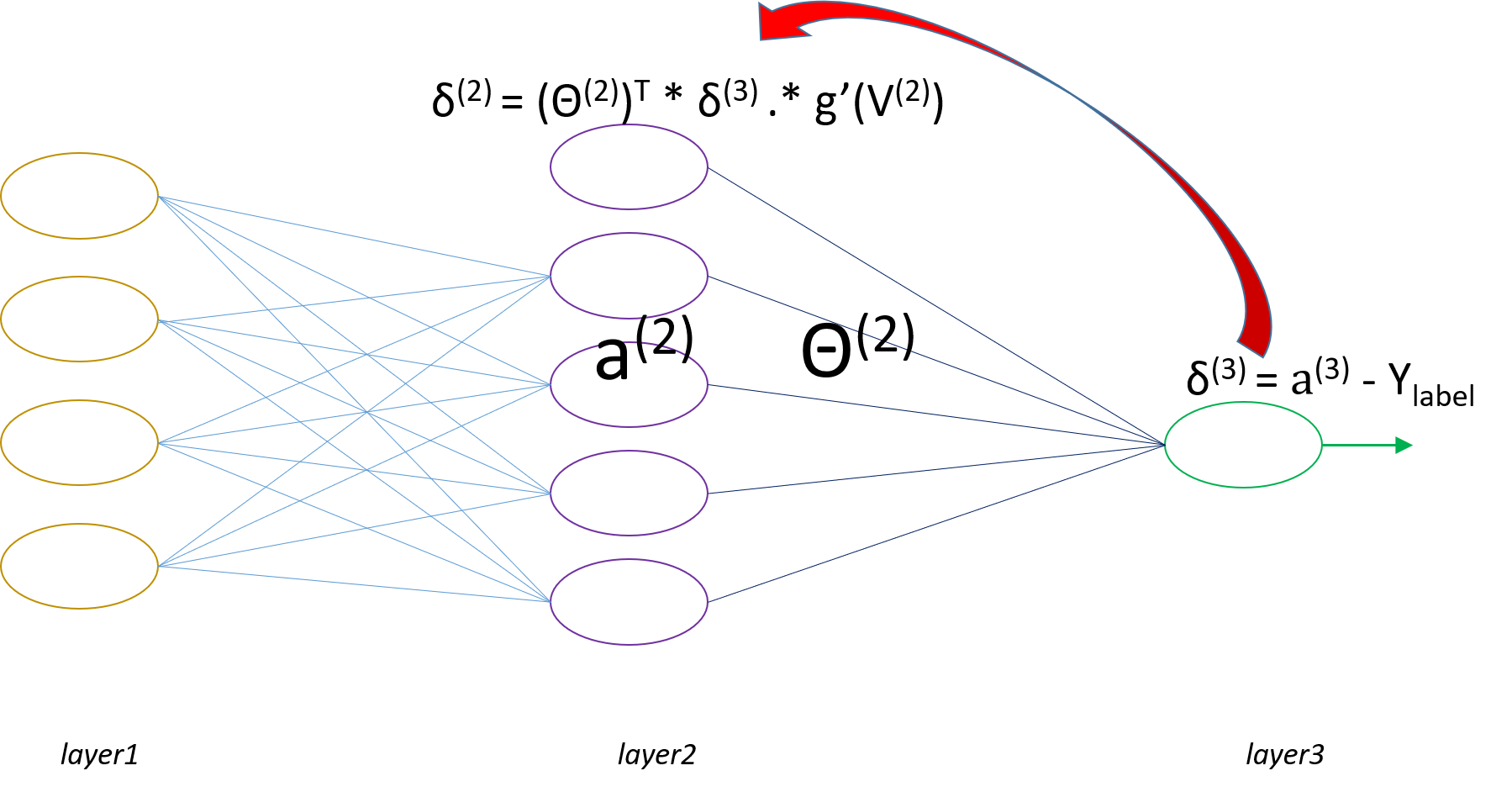
**חלחול לאחור -Backpropogation:**

אם נסכם מה שהיה לנו עד כה, אנחנו יודעים שרשת נוירונים מורכבת משכבות. לשכבה הראשונה נכנסות הדגימות מסט הנתונים. כל הדגימות עוברות זו אחר זו ומחושבת להן תוצאת ניבוי ומחושבת השגיאה ביחס לתוצאת האמת.

לאחר שכל הדגימות עברו ברשת, בודקים את פונקציית השגיאה, בעזרת שיטת ה -Gradient Descent מחליטים לאיזה ערך חדש לעדכן את המשקולות.

כלומר, יש בידינו את המשוואה לעדכון המשקולות של נוירון מסוים בהינתן מוצאו והמוצא הדרוש- ערך תוצאת האמת. עם זאת, במהלך אימון הרשת ידועה לנו רק שכבת היציאה של הרשת כולה, שכבת היציאה, והערך הרצוי לה. אנחנו לא ידועים את המוצאים והערכים הרצויים של שכבות הביניים. אלגוריתם Back-propagation הוא שיטה שמאפשרת לעדכן את המשקולות של שכבות הביניים גם כן - שכבה אחרי שכבה בתהליך של חלחול לאחור, החל משכבת הפלט ועד שכבת הכניסה.

****באיורים מטה מוסבר התהליך:





And we get the following connection:

= a(l)j\* δi(l+1(

**סיום תהליך הלמידה – סיכום השלבים**

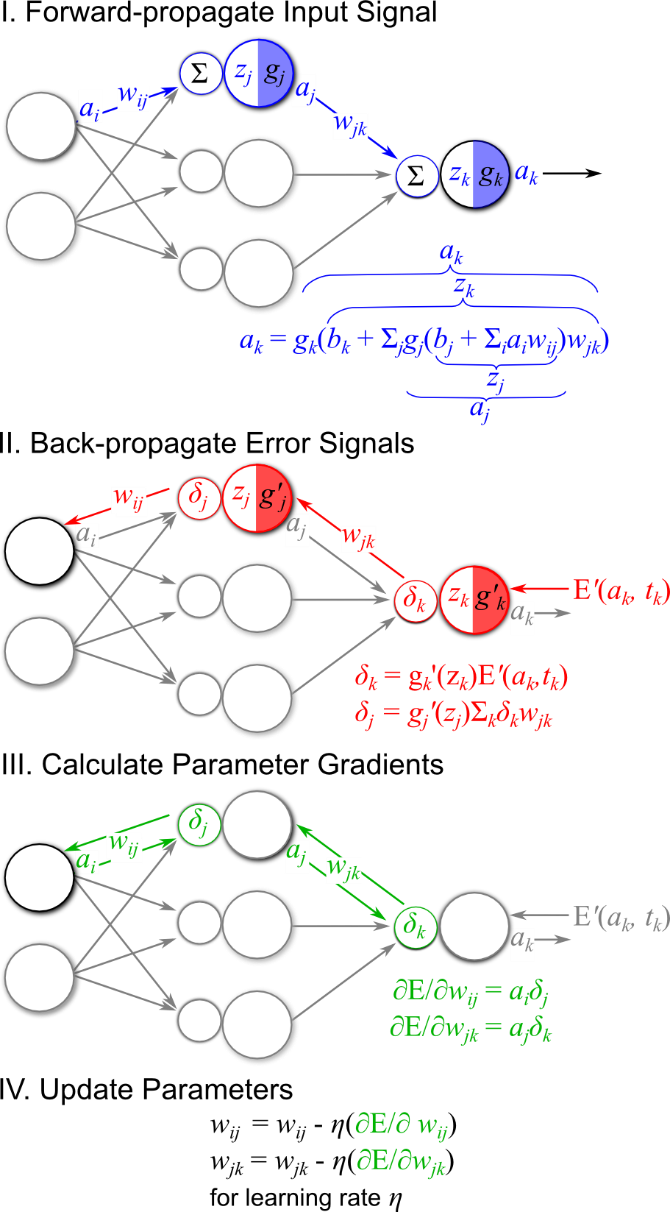
ביישומים פרקטיים נגביל בדרך כלל את כמות האיטרציות שמבצעת הרשת. נוכל לבחור יעד שגיאה מוגד מראש וכאשר היעד יושג, להפסיק את תהליך הלימוד. לא מובטח לנו שנגיע ליעד ולכן נגביל בכל מקרה את הכמות האיטרציות שאנו מאפשרים לרשת.

כאשר הרשת סיימה את תהליך הלימוד, קיבלנו את מודל החיזוי שלנו. עכשיו נוכל לבדוק את מידת הדיוק שלו אל מול סט מבחן, שעבורו אנו יודעים גם את תוצאות האמת.

כמובן נוכל גם להכניס דגימה חדשה, עבורה איננו יודעים את תוצאות האמת ולקבל עבורה חיזוי.

**סיכום בניית הרשת, האימון והשימוש:**

****



**חלק א.2 – רשתות נוירונים מלאכותית -פרקטיקה**

בשלב זה נתאר את החלק המעשי. אנחנו בחרנו בבעיית סיווג מיילים למייל ספאם ולמייל לגיטימי. זוהי משימה שיש לה שימושים פרקטיים בחיי היום יום. אנחנו נרצה לייצר מודל חיזוי שבהיתן מידע על מייל מסוים, יכריע אם המייל או ספאם או לא.

כדי ליצור מודל כזה, יישמו רשת נוירונים ע"י כתיבת תוכנית. התוכנית נכתבה בשפת פייתון.

חשוב לציין ששפת פייתון משמשת רבות בעולמות של בינה מלאכותית, למידה חישובית, אנליזה של כמויות מידע גדולות (big data) ועוד.

מאחר והשפה הזו כל כך פופולרית לשימושים האלה, ישנם ספריות מוכנות שמסייעות במימוש ושימוש באלגוריתמים שונים של למידה חישובית, בין היתר גם ברשתות נוירונים.

בין הספריות השונות ניתן למנות את tensor flow, keras, scit-learn,tflearn,theano ועוד רבות אחרות שהופכות את תהליך בניית הרשת, לעיתים לשורות קוד בודדות.

מאחר והיה חשוב לנו להבין ולממש את התהליך בעצמנו, ולהימנע מהאבסטרקציה וגישת ה- black-box, החלטנו לא להשתמש בספריות מוכנות ולכתוב את המימוש בעצמינו.

כפי שראינו בחלק התאורטי, תנאי הכרחי לאימון הרשת, הוא קיום של סט נתונים איכותי הכוללים תוויות סיווג שעליו תוכל הרשת להתאמן ולפיו לבסס את מודל החיזוי.

לצורך כך, השתמשנו במאגר מוכן הכולל 4601 דגימות (מיילים) שכל אחד מהם מסווג בינארית כספאם או לא ספאם. לכל מייל 57 תכונות שמאפיינות אותו, ביניהן אורך המייל, שכיחויות של מילים שונות שהוגדרו מראש ומופיעות בו ועוד.

כדי לאפשר הצגה גרפית של הנתונים, השתמשנו בטכניקה הנקראת PCA. PCA זו שיטה מתמטית שמאפשרת למצוא ייצוג ממימד נמוך יותר למידע רב מימדי, תוך שמירה מרבית על המידע שבמימד הגדול. כלומר ייצוג המידע במימד קטן יותר במינימום אבדן של ידע.

באמצעות שיטה זו הקטנו את הבעיה מבעיה של 57 מאפיינים לבעיה של 3 מאפיינים בלבד.

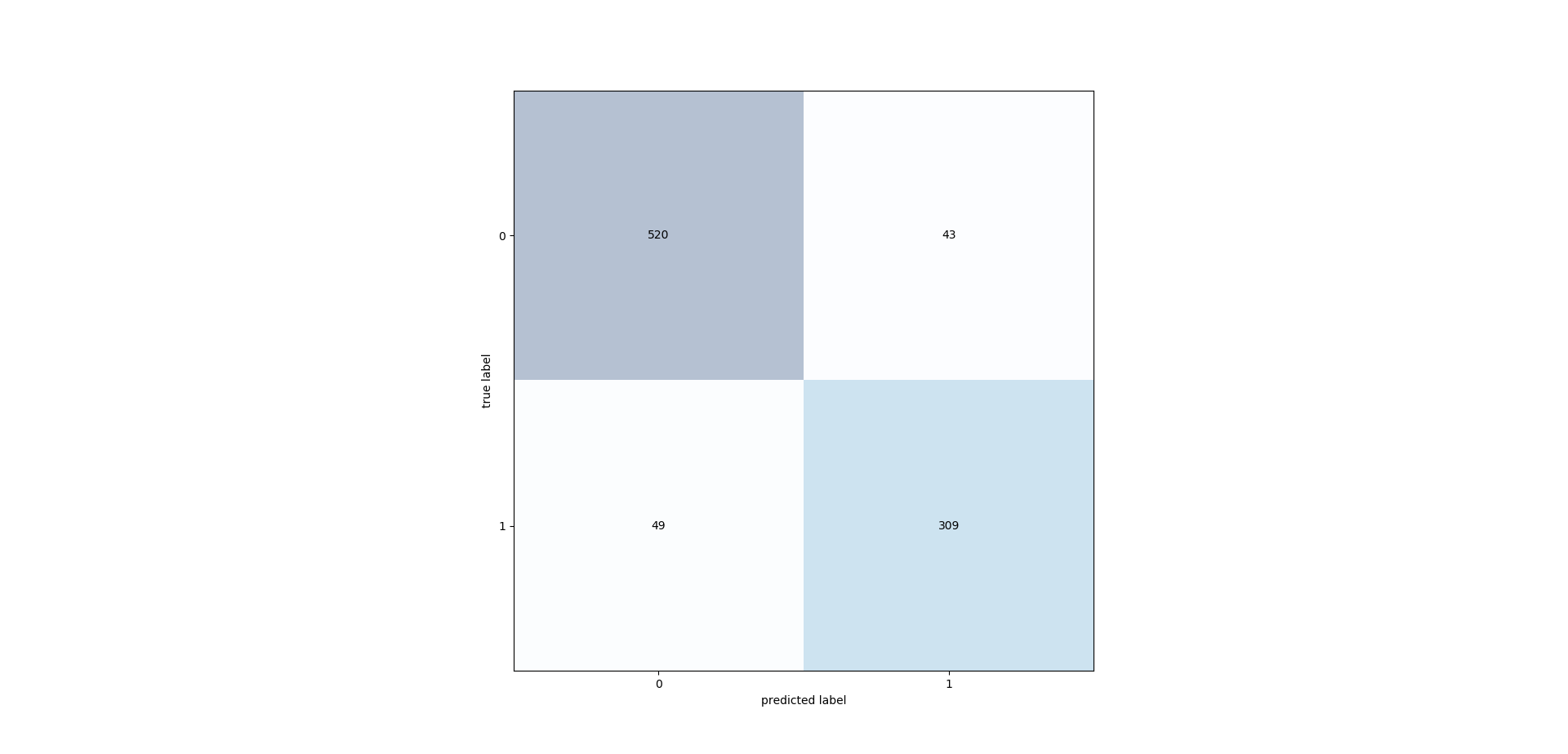
בנינו רשת נוירונים למידע המוצג במימד נמוך. שכבת הכניסה בגודל4 ( 3 התכונות + עמודה של bias), שכבה חבויה בגדול 5 ושכה פלט בגודל 1.

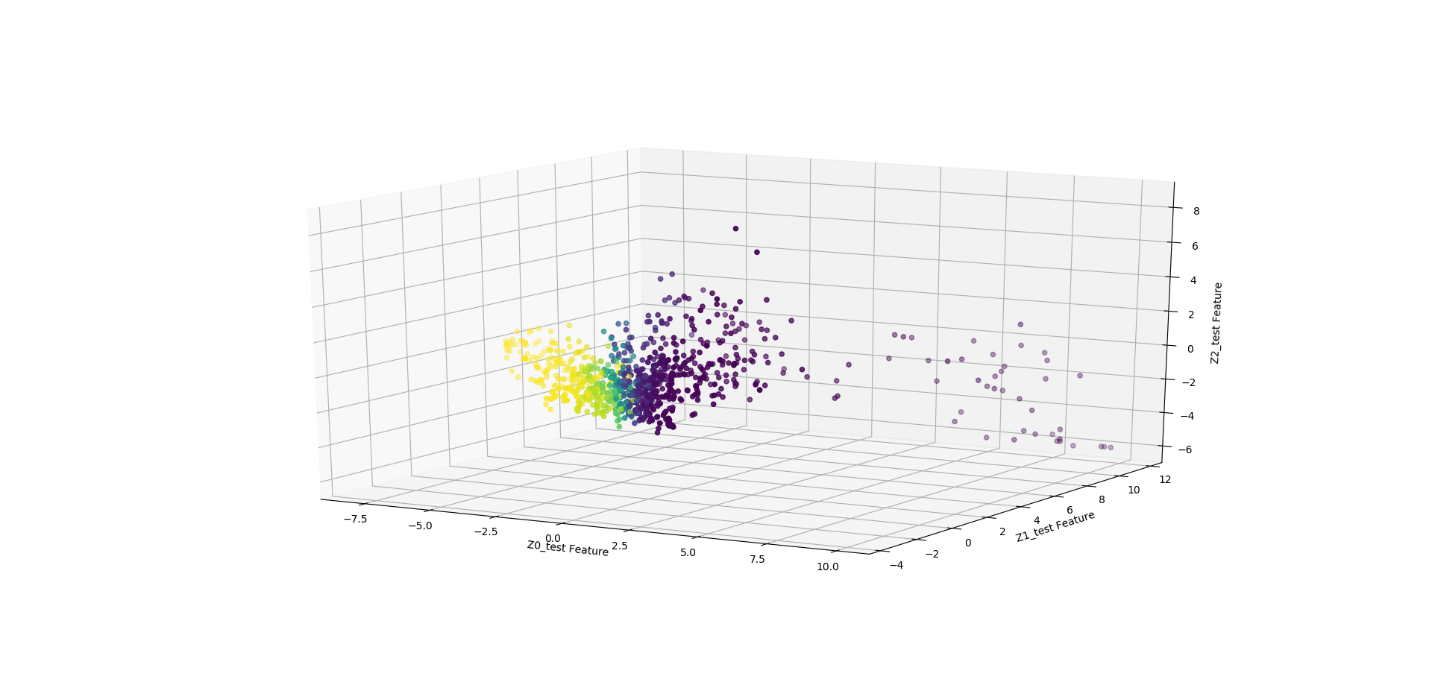
פונקציית האקטיבציה בה השתמשנו היא פונקציית הסיגמואיד, שנותנת ערך הסתברותי.

קבענו ערך סף 0.5, כך שעבור תוצאה עם ערך גבוה יותר, נסווג את המייל כספאם ואחרת נסווג כלא ספאם.

הרצנו סט מבחן עבור המודל שהתקבל והצגנו מטריצת בלבול (Confusion matrix) להצגת מידת ההצלחה של המודל. הצגנו את דיוק המודל, שחושב כחלק היחסי של החיזויים המוצלחים מסך כל החיזויים ובנוסך הצגנו פלט גרפי של תוצאות הסיווג.

מטריצת הבלבול- מציגה מה המודל חזה כספאם ואכן היה ספאם(True positive). מה חזה כלא ספאם ואכן לא היה ספאם (True Negative). מה חזה כספאם ודווקא לא היה ספאם(False positive). מה חזה כלא ספאם ודווקא כן היה ספאם(False Negative). אחוז ההצלחה בחיזוי עבור הרצה זו: 90.01%



פלט של פיזור המידע המסווג במרחב. אפשר לראות הפרדה ברורה בין הספאם והלא ספאם, מה שמעיד על הצלחה טובה של המודל בסיווג.

**חלק ב.1 – אלגוריתם K מרכזים (Kmeans) -התאוריה:**

**רקע ללמידה חישובית בלתי מונחית (Unsupervised Learning):**

בחלק א' דיברנו על רשת נוירונים מלאכותית שהיא אלגוריתם המשתייך למשפחת האלגוריתמים של למידה מונחית, ובהקדמה לתיאור האלגוריתם תיארנו מהי למידה מונחית.

בחלק זה נדבר על אלגוריתם Kmeans שמתייך למשפחת האלגוריתמים של למידה לא מונחית ולכן נקדים ונפתח בהסבר על מהי למידה לא מונחית.

ההבדל המשמעותי ביותר בין למידה מונחית אותה פגשנו, לבין למידה שאינה מונחית הוא בסוג הנתונים שמביאים לאלגוריתם כקלט. כמובן, סוג המידע קשור קשר הדוק לאופן שבו האלגוריתם פועל.

בלמידה מונחית, לכל דגימה בסט הנתונים הייתה תווית סיווג (לייבל). האלגוריתם למד בהתאם לסיווג, יצר מודל ולאחר מכן, עבור דגימה חדשה, נתן סיווג שהוא החיזוי.

באלגוריתמים השייכים למשפחת של הלמידה הלא מונחית, המידע שמתקבל, ניתן ללא תווית. כלומר, המידע לא מגיע עם סיווג ולאלגוריתם אין דרך להשוות או לדרג פתרונות אפשריים. המטרה של האלגוריתם אם כך, היא למצוא מאפיינים בתוך המידע, דפוסים ותבניות, סידור טבעי, ולמעשה לייצר את הסיווג בעצמו.

אלגוריתמים ממשפחת הלמידה הבלתי מונחית משמשים פעמים רבות למשימת ניתוח אשכולות. מאחר וזו גם אחת המטרה באלגוריתם kmeans נרצה להבין את משמעות המושג.

**ניתוח אשכולות (חלוקה לקלאסטרים):**

ניתוח אשכולות הוא מושג המשמש בתחומים של כריית מידע, סטטיסטיקה ולמידה חישובית. ניתוח אשכולות Cluster Analysis)) מתייחס למשימה של קיבוץ אובייקטים לקבוצות (אשכולות) הנקראים פעמים רבות קלאסטרים, כך שהאובייקטים הנמצאים באותה קבוצה דומים זה לזה יותר מאשר לאובייקטים השייכים לקבוצות אחרות.

לדוגמא, ניקח את קבוצה של אנשים, ונחלק אותם לתתי קבוצות על בסיס המאפיינים שלהם. קל להיווכח שאפילו בסט האובייקטים הכי פשוט נמצא מאפיינים רבים שעל פיהם ניתן היה לקבץ את האובייקטים בקבוצות שונות, כאשר התכונות ייבחרו כמובן בתלות במטרה של הניתוח המידע המבוקש. משתמשים בניתוח אשכולות בתחומים שונים בבינה מלאכותית, בזיהוי תבניות, ניתוח תמונה, ועוד יישומים רבים אחרים.

ישנם סוגים רבים של אלגוריתמים שפותחו לניתוח אשכולות, בין היתר אלגוריתמים מבוססי קישוריות (שבבסיסם המרחק בין אובייקטים שונים), אלגוריתמים מבוססי התפלגות סטטיסטית או אלגוריתמים מבוססי תורת הגרפים.

**אלגוריתם Kmeans עיקרון האלגוריתם:**

אלגוריתם k-מרכזים(k-means) הוא שיטה פופולרית עבור ניתוח אשכולות (Clustering). מטרתו לחלק את התצפיות (סט הנתונים) ל-k אשכולות לפי מרכזי כובד . כל תצפית משויכת לאחד מ"מרכזי הכובד".

K-Means הוא האלגוריתם הנפוץ ביותר מסוג אלגוריתמים מבוססי מרחק.

אלגוריתם זה מבוסס על חילוק האובייקטים למספר של אשכולות לפי מרכזי כובד, כאשר אנחנו לא יודעים מי הם מרכזי הכובד שלנו בהתחלה.

בהינתן קבוצה של תצפיות כאשר כל תצפית היא וקטור ממשי היכול להיות בעל מספר מימדים,

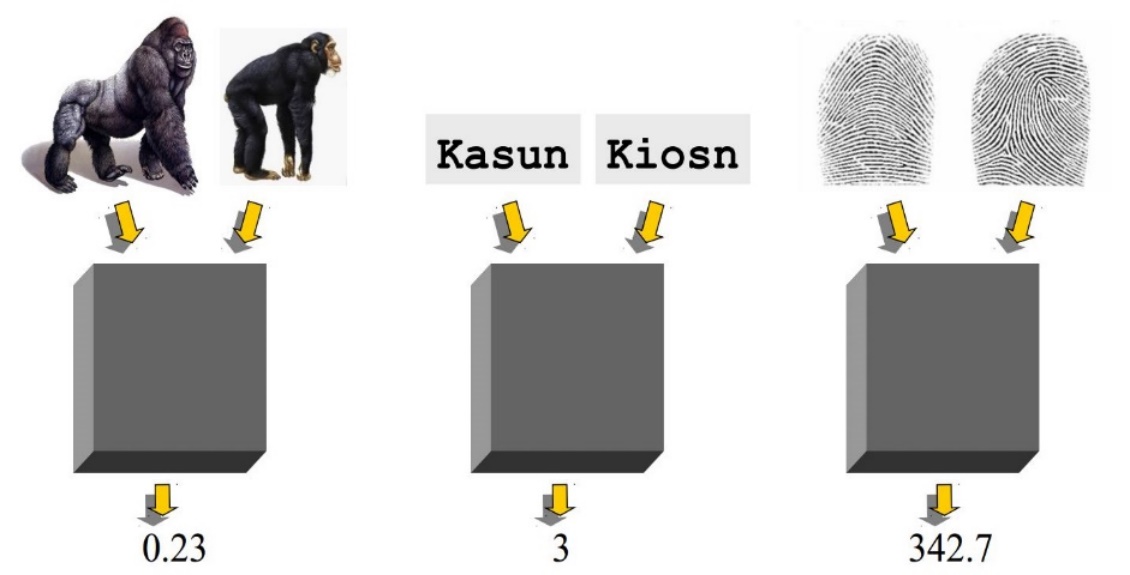
המודל שואף לחלק את n התצפיות לk אשכולות, על מנת למזער את סכום המרחקים בין התצפיות (הווקטורים) בתוך האשכול.

**המטריקה:**

הנתונים אותם אנו מקבלים כקלט ואותם נדרש האלגוריתם לחלק לאשכולות אינם בהכרח נתונים נומריים ויתרה מכך, הם יכולים להכיל אינפורמציה רבה על האובייקטים כשאר אנו מעוניינים למצוא קרבה בהתאם למאפיינים ספציפיים. מסיבה זו עלינו לבצע תחילה תהליך של עיבוד מקדים למידע.

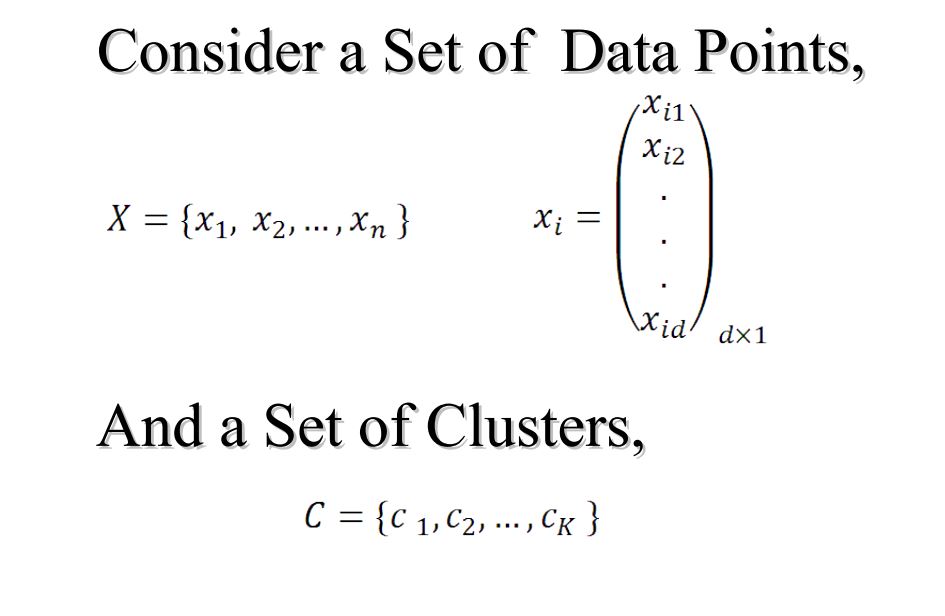
בתהליך זה נשאף לקבל מדד שלפיו נוכל להשוות מרחק בין אובייקטים. נניח נתונה לנו קבוצת אנשים ומידע אודות מאפיינים שונים שלהם ונרצה אפוא לחלק את האנשים לקבוצות טבעיות. האם נגיד ששני אנשים הם קרובים זה לזה, כלומר צריכים להשתייך לאותה קבוצה, אם יש להם אותו גובה ומשקל, או אולי בהתאם למקום המגורים שלהם, או תחומי העניין וכן הלאה וכן הלאה.

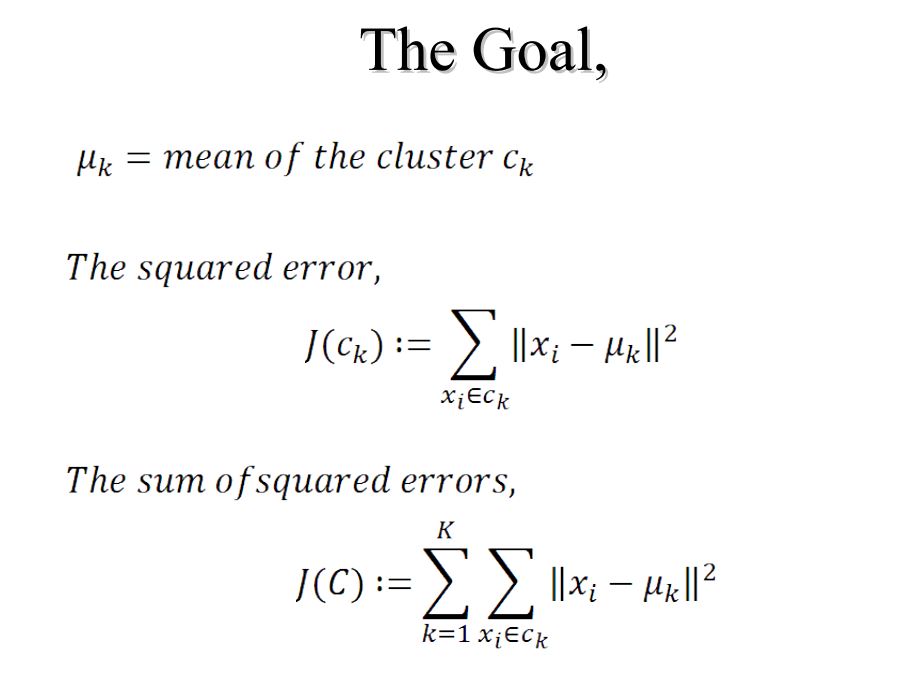
המרחק בין אובייקטים הוא סובייקטיבי ותלוי בעולם הבעיה ובמטרה של שלשמה הופעל האלגוריתם. המטריקה היא פונקציה המתאימה לכל זוג נקודות במרחב מספר אי-שלילי. למספר זה אנו מייחסים את מושג המרחק בין האובייקטים. המטריקה היא הכללה של מושג המרחק מהמרחב האוקלידי למרחב כלשהו. אנו נרצה לעשות המרה לנתונים המקוריים וליצור לנו קידוד כזה המאפשר לנו להתייחס לאיך שאנו מגדירים מרחק בעולם הבעיה לצורה המוכרת של נוסחת מרחק אוקלידי.



**הדגמת האלגוריתם**

מטרת האלגוריתם היא למצוא את מרכזי הכובד האופטימליים כך שסכום המרחקים הריבועיים בין כל נקודה ומרכז הכובד אליו שויכה, כלומר האשכול שלה, יהיה הקטן ביותר.





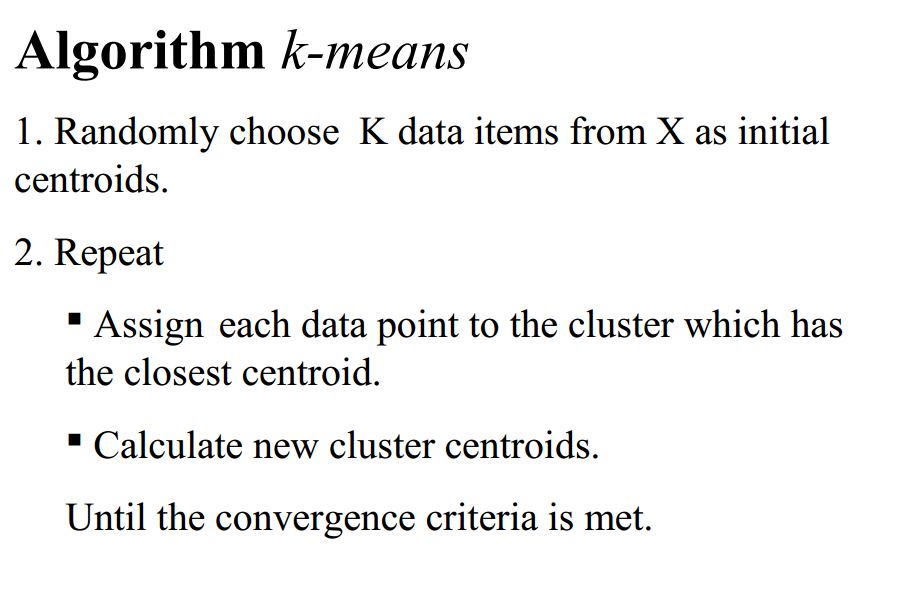
**ככלל, האלגוריתם ינוע לסירוגין בין שני שלבים הבאים:**

שלב הקצאה: עלינו להקצות לכל אובייקט את אחד האשכולים. בהמשך נראה כיצד אנו מחליטים לאיזה אשכול ישויך כל אובייקט

שלב עדכון: עלינו לחשב את מרכזי הכובד החדשים כדי להיות במרכז הכובד של האובייקטים באשכולות החדשים.

האלגוריתם עוצר כאשר לא ניתן לשנות עוד את הנתונים, ומתקבל ערך קבוע של מרכזי כובד, כלומר כאשר הגיע להתכנסות, או בהתאם למספר איטרציות שהוגדר מראש. חשוב לשים לב שהאלגוריתם אינו חייב להתכנס, ולכן בכל אופן יש להגדיר כמות איטרציות ידועה מראש. ניתן להשתמש גם במדד שגיאה כתנאי עצירה נוסך לכמות האיטרציות. ניתן לחשוב על מצב שהאלגוריתם אינו מתכנס כאשר יש אובייקט שקרוב במידה שווה לשני מרכזי כובד (אשכולים) ונע לסירוגין ביניהם במהלך הריצה.

**שלבי האלגוריתם:**

1. נבחר באקראי k אובייקטים שיהוו את מרכזי הכובד מתוך האובייקטים שניתנו כקלט התחלתי.
2. נשייך כל נקודה למרכז הקרוב אליה ביותר, על ידי פונקציית המרחק בין נקודות (פונקציית מרחק אוקלידי פשוטה).
3. מעדכנים כל אשכול (מרכז כובד) להיות הממוצע של כל הנקודות ששויכו לאשכול, על ידי חישוב המרחקים בין כל הנקודות וחלוקה במספר הנקודות לקבלת ממוצע.
4. נחזור על צעדים 2-3, עד שאין שינוי באף אחד ממרכזי הכובד או עד כמות איטרציות ומדד דיוק ידועים מראש.

**זמן ריצה:**

זמן ריצת האלגוריתם תלוי בגורמים הבאים:

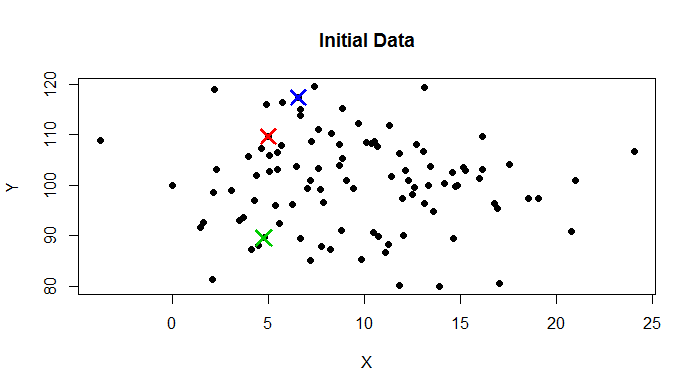
N –מספר האובייקטים ההתחלתי (גודל הקלט)

K – מספר האשכולות הרצוי (K שנבחר מראש)

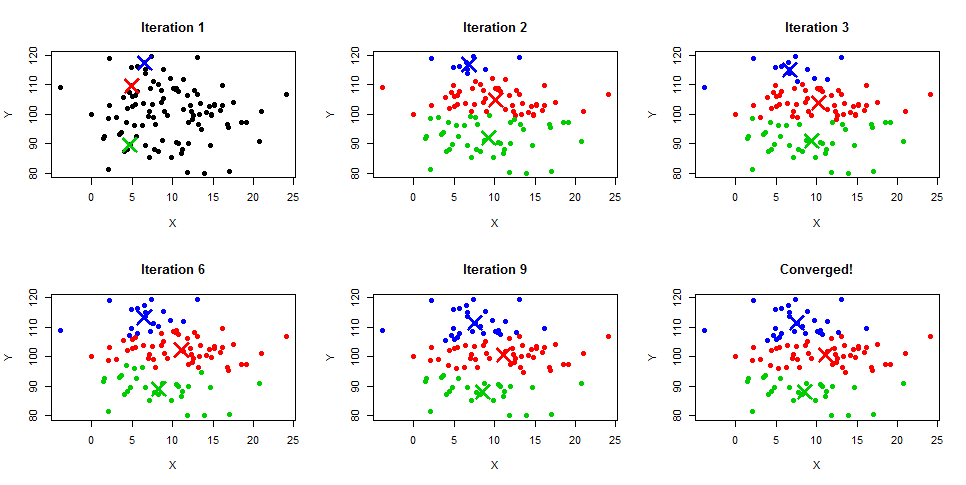
I – מספר האיטרציות

D – מספר המאפיינים על המידע (כמות הפיצ'רים/תכונות שמתארות נקודה/ דגימה בקלט)

זמן ריצת האלגוריתם היא O(n \* k \* i \* d)

**דוגמה לריצת האלגוריתם:**

* איטרציה מספר 2 מראה את המיקום החדש של מרכז הכובד
* באיטרציה מספר 3 יש כמות גדולה משמעותית של נקודות כחולות כאשר מרכז הכובד הוזז
* בקפיצה לאיטרציה מספר 6, רואים שמרכז הכובד האדום זז רחוק יותר לימין
* באיטרציה מספר 9 רואים שהחלק הירוק הוא קטן משמעותית מאשר באיטרציה מספר 2. הכחול השתלט על החלק העליון, ומרכז הכובד האדום דל יותר מאשר היה באיטרציה 6.
* תוצאות האיטרציה ה-9 זהות לאיטרציה מספר 8, כלומר המודל התכנס.



**חלק ב.1 – אלגוריתם K מרכזים (Kmeans) -פרקטיקה:**

המשימה שבחרנו ליישם היא דחיסת תמונה. אנחנו נשתמש במימוש של האלגוריתם Kmeans ונראה כיצד אפשר לנצל את האלגוריתם היחסית פשוט ואת הצורה שבה פועלת דחיסה של קובץ PNG כדי לדחוס תמונה תוך שמירה על המראה הכללי של התמונה.

דחיסה של תמונה בפורמט PNG היא דחיסה ללא עיוות (Lossless compression). משמעות הדבר שאינטנסיביות הצבעים, הכמות שלהם, גדול התמונה ובאופן כללי איכות התמונה לא נפגעת.

יחד עם זאת, אם נפחית את כמות הצבעים בתמונה המקורית, נוכל ליצור תמונה ששומרת באופן כללי על צורתה של התמונה המקורית, אך נפחה קטן משמעותית. כלומר הקטנת כמות הצבעים הייחודים בתמונה, מאפשר לPNG לדחוס את התמונה לגודל קטן יותר. שאר המאפיינים כגון גודל התמונה, ואינטנסיביות הביטים אינה משתנה. המונח לכמות הצבעיים הייחודים בתמונה נקרא Palette והתהליך של הפחתת הצבעים נקרא Color quantization.

כלומר, התהליך שנעשה יניב דחיסה עם עיוות, כלומר תוצר הדחיסה אינו זהה למקור, אך המטרה היא שהנראות הכללית תישמר כך שחווית הצפייה בתמונה לא תיפגע באופן משמעותי.

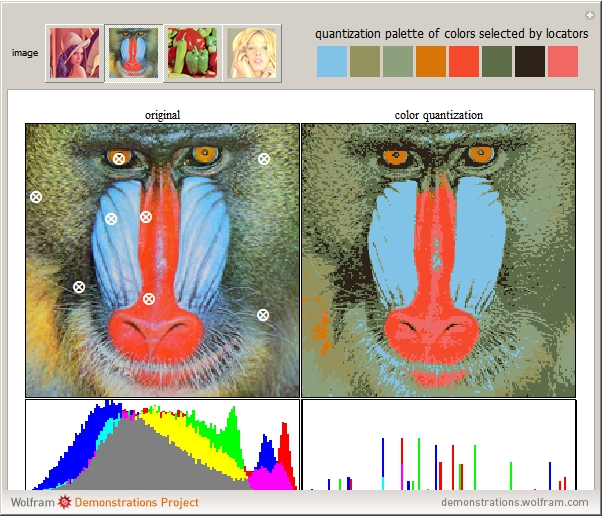
השתמשנו בספרייה מכונה שמממשת את האלגוריתם. באלגוריתם מסוג זה האתגר אינו היישום, מאחר שמדובר בסה"כ בחישוב נוסחת מרחק אוקלידי בסיסי. המטרה שלנו הייתה להראות כיצד ניתן להשתמש באלגוריתם פשוט לבצע פעולה שנראה מורכבת יחסית, כמו דחיסה של תמונה.

**עיקרון הפעולה:**

בדחיסת תמונה ננצל את אופן הדחיסה של PNG ואת היכולת של K-means למציאת מרכזי כובד.

הרעיון הוא כאמור לקחת תמונה, וליצור ממנה תמונה חדשה, שמתבססת על התמונה המקורית אך עם כמות צבעים (palette) קטנה יותר תוך שמירה על הנראות הכללית של התמונה.

העיקרון- כל מרכז כובד מייצג צבע ייחודי. כלומר K יקבע את מספר הצבעיים בתמונה הדחוסה. האלגוריתם ישייך כל פיקסל בתמונה המקורית למרכז שאליו הוא הכי קרוב בבחינת מרחק צבעים.



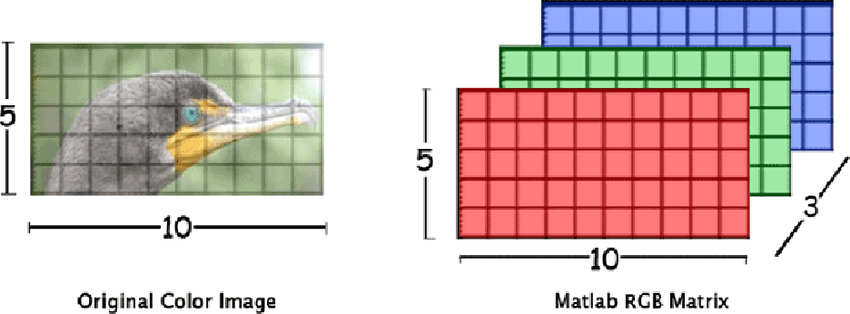
דוגמה לדחיסת Color quantization

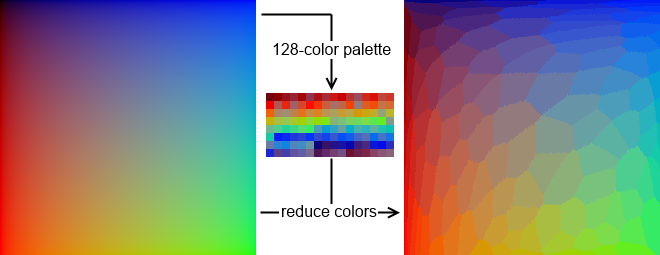
כל תמונה ניתנת לייצוג ע"י מטרית RGB (כמתואר באיור)

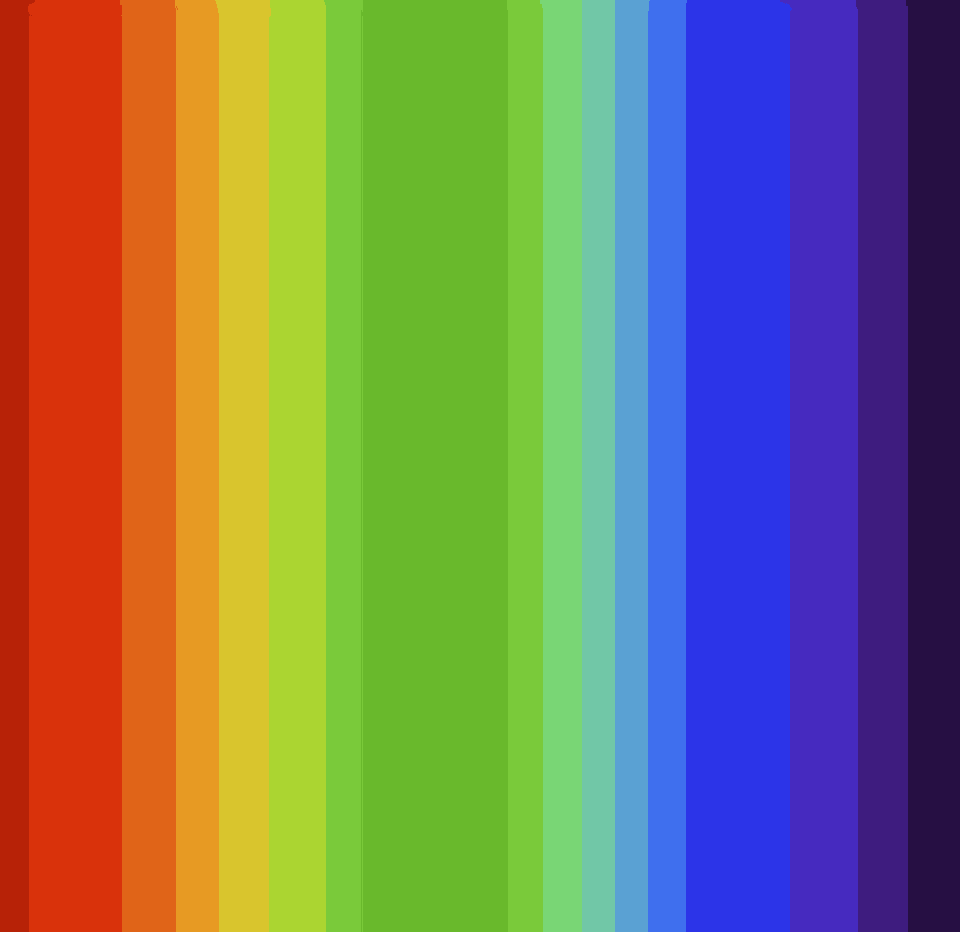
אנו נייחס למוגש מרחק את המרחק בין הצבעים:



זוהי המטריקה שלנו שזהו בעצם מרחק אוקלידי במרחב, שבו כל נקודה היא ערכי הפיקסל בהצגת RGB.



**הדגמת התוצאות- נראות ודחיסה:**

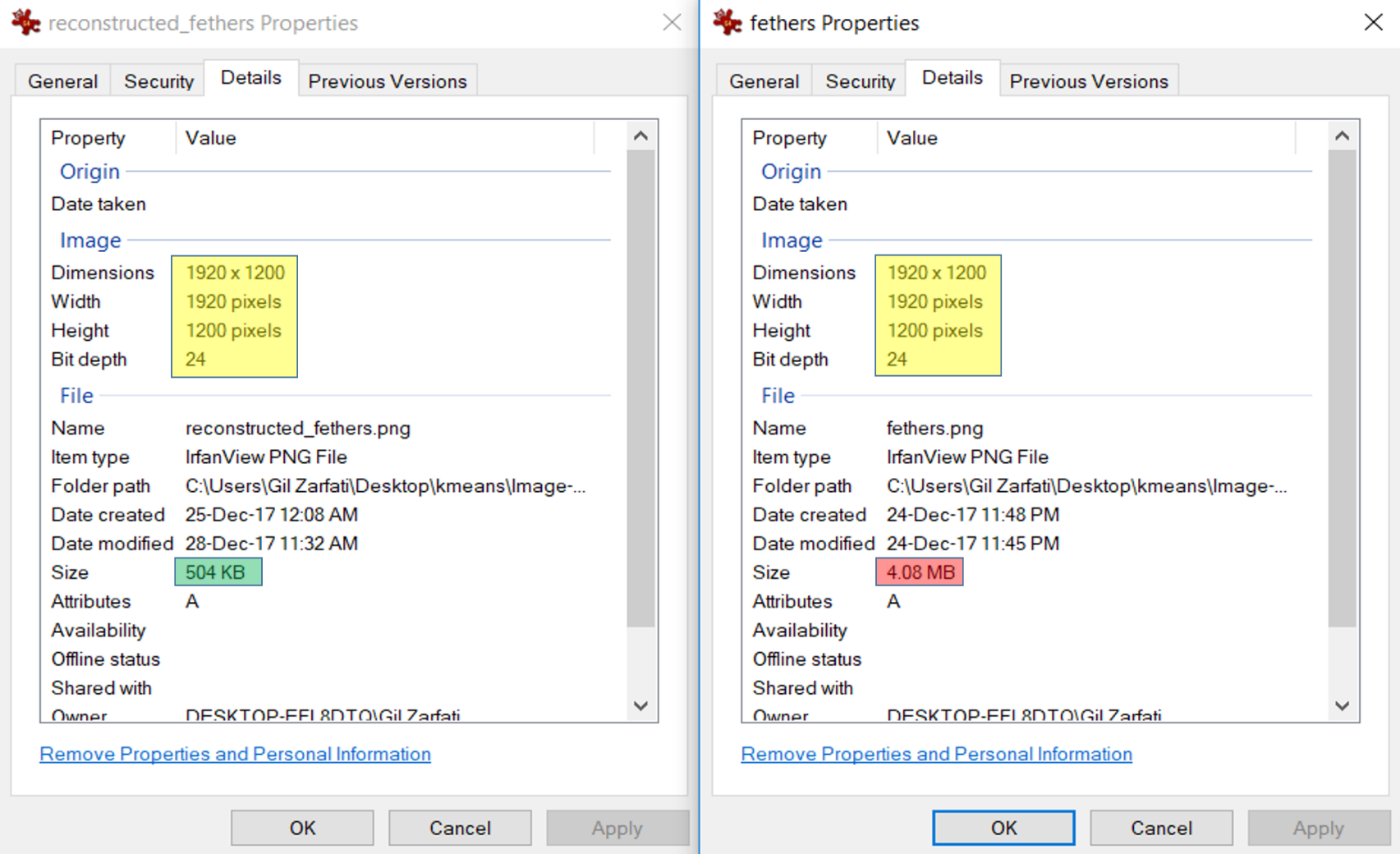
התוצאות הנ"ל התקבלו מהרצת האלגוריתם:







נסתכל בפרטי הקובץ של שתי התמונות של הנוצות. נשים לב כי כל הפרטים זהים, למעט נפח התמונה.



נספח א- קוד לרשת נוירונים

נספח ב- קוד KMEANS

**ביביליוגרפיה:**

<https://en.wikipedia.org/wiki/Artificial_neural_network>

<https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471>

<https://en.wikipedia.org/wiki/Cluster_analysis>

<https://www.udemy.com/machinelearning/learn/v4/overview>

<https://machinelearnings.co/text-classification-using-neural-networks-f5cd7b8765c6>

<https://appliedgo.net/perceptron/>

<https://en.wikipedia.org/wiki/Color_quantization>

<https://en.wikipedia.org/wiki/K-means_clustering>

<https://www.pyimagesearch.com/2014/05/26/opencv-python-k-means-color-clustering/>

<http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

נספח א- קוד לסיווג מיילים ע"י רשת נוירונים



נספח ב- קוד לדחיסת תמונה באמצעות K-means

