בס"ד

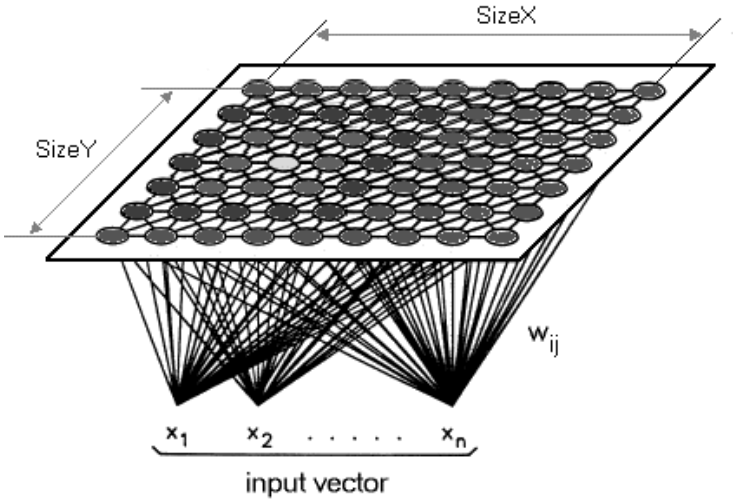
ביולוגיה חישובית – SOM

דו"ח זה הינו עבור אלגוריתם SOM לסיווג קבוצות על דאטה (במקרה שלנו וקטורי מספרים) לנוירונים מייצגים.

בכללות, **אלגוריתם SOM** בנוי ממספר שלבים פשוטים:

1. יצירת ואתחול קבוצת נוירונים
2. מעבר על כלל הדאטה ומציאת BMU עבור כל וקטור
3. קירוב כל וקטור BMU לוקטור מהדאטה המתאים לו ואת שכניו

וחוזר חלילה על שלבים 2-3 עד עצירה.



דו"ח זה מחולק לשבעה חלקים:

א. רשת הנוירונים
ב. וקטור ה- BMU
ג. עדכון השכנים
ד. learning rate
ה. טיב הפתרון
ו. היפר-פרמטרים
ז. הוראות הרצה

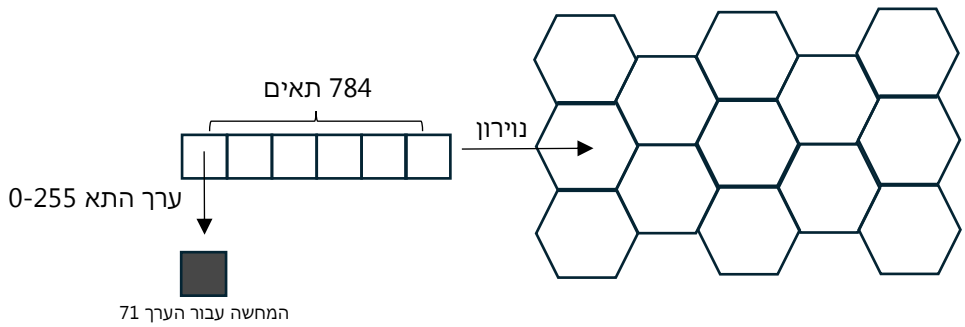
חלק א' – רשת הנוירונים

ייצוג הנוירונים:

כאשר ממשים רשת SOM, יש לשמור על הנוירונים במבנה נתונים, אותו מעדכנים במהלך הריצה של האלגוריתם. מבנה הנתונים בו בחרנו הינו רשת העשויה ממשושים, כאשר כל איבר ברשת הינו וקטור בגודל 784, כגודל תמונה 28x28, ובתוך כל תא בוקטור הנוירון ישנו ערך בין 0-255 המייצג צבע בצבעי שחור-לבן.

לשם פישוט השימוש במשושים יצרנו מחלקה בשם: HexBoard, בה מתרחש כלל הטיפול בניהול מבנה הנתונים של רשת הנוירונים.

נדגים על ידי תרשים:



אתחול הנוירונים:

ישנן שיטות אתחול רבות: אתחול רנדומי, אתחול לאפסים ועוד. אנו בחרנו לאתחל כל נוירון באופן מעט שונה מהרגיל.

את אתחול הנוירונים שלנו עשינו באופן מעט שונה. ביצענו מעבר על כלל הדאטה אותו קיבלנו בצורה איטרטיבית, לקחנו בכל פעם דגימה לכל נוירון, סכמנו אותן (element wise) וחישבנו עליהם ממוצע. סך הכול יצרנו 100 וקטורי ממוצע כנ"ל, אותם הגדרנו להיות וקטורי הרשת – הנוירונים.

סה"כ נקבל - עבור N_i הנוירון ה- i, u_j הוקטור ה- j בדאטה סט:

$$\forall 1 \le i \le 100, \forall 1 \le j \le 10000, j\%100 = i:\ N_i = \frac{\sum_j u_j}{100}$$

הסיבה בה בחרנו לאתחל דווקא באופן זה, הינה כי לרוב ישנם מאפיינים משותפים לדאטה, ובצורה כזו אנחנו יכולים לתפוס אלמנטים כאלו כבר בהכנת הרשת. למשל, במקרה שלנו, לכלל התמונות יש מסגרת שחורה (רצף אפסים) , כאשר נחשב ממוצע על מספר דגימות נקבל כי הנוירון המקבל את ערך הממוצע הנ"ל יהיה גם בעל המאפיין של מסגרת שחורה. בנוסף, אופן הסכימה המעגלית, מייתר את ההישענות על כמות מסוימת של דאטה המתחלקת בצורה מושלמת.

כמות הנוירונים ברשת:

כשם שציינו לעיל, כמות הנוירונים ברשת הינה 100. במקרה שלנו בו ישנם 10 ערכים אפשריים, זהו מספר הגיוני, כיוון שכך אנו מייצגים כעשרה נוירונים עבור כל ספרה, נותנים מקום לאלמנטים שונים בכל ספרה ומפרידים בצורה טובה יותר בין וריאנטים שונים.

חלק ב' – וקטור ה- BMU

בכל איטרציה של ריצת האלגוריתם, אנו עוברים על כלל האיברים בדאטה, מוצאים עבור כל איבר BMU ומקרבים את ה- BMU לאותו האיבר.

ראשית, נסביר מעט מה זה בכלל BMU.

BMU - Best Matching Unit, הינו וקטור הנוירון מהרשת אשר "קרוב" ביותר לוקטור הדאטה אותו אנו דוגמים בכל פעם.

הגדרת הקרבה בין וקטורים ניתנת לפירוש ומימוש בצורות שונות, אנו בחרנו להגדיר קרבה בפשטות על ידי מימוש בדיקת מרחק אוקלידי, עבור N_i הנוירון ה- i, u_j הוקטור ה- j בדאטה סט:

$$BMU = \underset{i}{\operatorname{argmax}} \left\{ \sqrt{\sum_{j=0}^{10000} \left(N_i + u_j\right)^2} \right\}$$

כעת, לאחר שהסברנו מה הוא ה- BMU, נתבונן בתהליך העדכון:

```
# Function to update the neurons using the dataset
def update():
    global NEURONS_MATRIX

    for vector in DS:
        idx = extract_bmu_idx(vector)
        i, j = idx
        NEURONS_MATRIX[i][j] = calc_update(vector, NEURONS_MATRIX[i][j], LEARNING_RATE)
        neighbors_update(vector, idx)
```

1. בצע לולאה על כל הוקטורים בדאטה סט:

a. חלץ את האינדקסים של ה- BMU עבור הוקטור.
b. הפרד את האינדקסים של ה- BMU ל- i ו- j.
c. עדכן את ערך הנוירון (BMU) ברשת.
d. עדכן את השכנים של ה- BMU (פירוט בחלק ג')

כעת, נסביר את הרעיון מאחורי הלוגיקה.

כשם שהסברנו לעיל, אנו מבצעים מעבר מלא על כלל הדאטה סט, ועבור כל וקטור מהדאטה אנו מוצאים את הנוירון הקרוב אליו ביותר (BMU) ומקרבים את אותו הנוירון לוקטור עוד יותר.

ביצוע ה"קירוב" הינו חישוב המרחק ביניהם בכל תא, והוספת חלק מאותו פער ל- BMU על מנת שיהיה דומה יותר לוקטור מהדאטה סט (ערכי העדכון המדויקים ופירוט עליהם בחלק ד'). בנוסף, לאחר עדכון ה- BMU, נבצע עדכון וקירוב (קל יותר) אף של שכניו (פירוט בחלק ג').

כך אנו דואגים שכל נוירון ברשת המייצג מספר איברים מסוים יהיה קרוב אליהם ברמה מסוימת, דבר המאפשר לאחר סיום בניית הרשת, סיווג נכון יותר של וקטורים חדשים.

חלק ג' – עדכון השכנים

בחלק הקודם ביארנו על ה- BMU, וציינו כי כחלק מתהליך העדכון שלו אנו מעדכנים אף את שכניו.

ברשת הנוירונים אותה אנו בונים במודל SOM, ישנה משמעות אף למיקום נוירונים שונים ברשת (פירוט בחלק ה') – אנו נרצה שנוירונים דומים יהיו קרובים אחד לשני. לשם כך, כאשר אנו מקרבים נוירון כלשהו לוקטור מהדאטה, נרצה לקרב אף שכניו לאותו הוקטור.

נשים לב, שאומנם אנו רוצים לקרב את שכניו גם כן, אבל נעשה זו בזהירות, כיוון שלא נרצה לשבש את כלל הרשת בכל צעד, נוריד את רמת העדכון בכל פעם במעט ואת עדכון השכנים נבצע בעוצמה קלה יותר (ערכי העדכון המדויקים ופירוט עליהם בחלק ד').

נתבונן בתהליך עדכון השכנים:

```
# Function to update the neighbors of the BMU
def neighbors_update(vector, idx):
    global NEURONS_MATRIX

    first_learning_rate = LEARNING_RATE / (ITERATIONS + 1)
    second_learning_rate = first_learning_rate / (ITERATIONS + 1)
    i, j = idx

    first_layer = []
    second_layer = []
    first_layer_directions = []
    second_layer_directions = []

    if j % 2 == 1:
        # Define the directions for the first layer neighbors
        first_layer_directions = [(0, -1), (0, 1), (1, -1), (1, 0), (1, 1), (-1, 0)]
        # Define the directions for the second layer neighbors
        second_layer_directions = [(-2, 0), (-1, 1), (-1, 2), (0, 2), (1, 2), (2, 1), (2, 0), (-1, -1), (-1, -2), (0, -2), (1, -2), (2, -1)]

    if j % 2 == 0:
        # Define the directions for the first layer neighbors
        first_layer_directions = [(-1, -1), (-1, 0), (-1, 1), (0, 1), (0, -1), (1, 0)]
        # Define the directions for the second layer neighbors
        second_layer_directions = [(-2, -1), (-2, 0), (-2, 1), (-1, 2), (0, 2), (1, 2), (-1, -2), (0, -2), (1, -2), (1, -1), (1, 1), (2, 0)]

    for di, dj in first_layer_directions:
        ni, nj = i + di, j + dj
        if 0 <= ni < 10 and 0 <= nj < 10:
            first_layer.append((ni, nj))
```

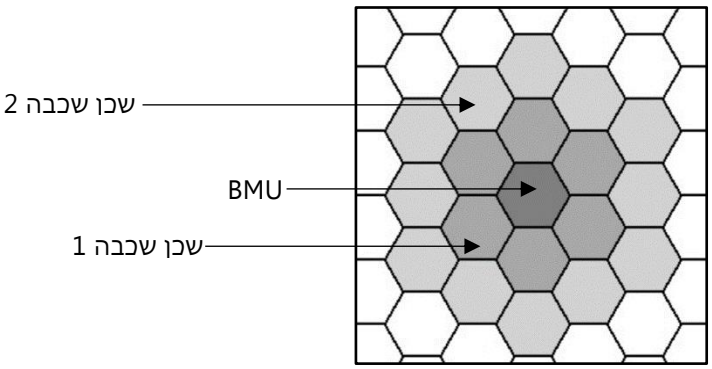
```
for di, dj in second_layer_directions:
    ni, nj = i + di, j + dj
    if 0 <= ni < 10 and 0 <= nj < 10:
        second_layer.append((ni, nj))

# Update first layer neighbors
for ni, nj in first_layer:
    NEURONS_MATRIX[ni][nj] = calc_update(vector, NEURONS_MATRIX[ni][nj], first_learning_rate)

# Update second layer neighbors
for ni, nj in second_layer:
    NEURONS_MATRIX[ni][nj] = calc_update(vector, NEURONS_MATRIX[ni][nj], second_learning_rate)
```

1. נאתחל learning rate עבור שכבת השכנים הראשונה.
2. נאתחל learning rate עבור שכבת השכנים השנייה.
3. נחלץ את המיקום של ה- BMU ונשמור ב- i ו- j.
4. נאתחל רשימה ריקה עבור איברי שכבת השכנים הראשונה.
5. נאתחל רשימה ריקה עבור איברי שכבת השכנים השנייה.
6. נאתחל רשימה ריקה עבור כיווני שכבת השכנים הראשונה.
7. נאתחל רשימה ריקה עבור כיווני שכבת השכנים השנייה.
8. נאתחל את כיווני שכבת השכנים עבור j אי זוגי וזוגי בהתאמה.
9. בצע לולאה על כל כיווני השכבה הראשונה:
    a. חשב אינדקס שכן.
    b. אם האינדקסים חוקיים, הוסף שכן זה לשכבת השכנים הראשונה.
10. בצע לולאה על כל כיווני השכבה השנייה:
    a. חשב אינדקס שכן.
    b. אם האינדקסים חוקיים, הוסף שכן זה לשכבת השכנים השנייה.
11. בצע לולאה על כל לשכבת השכנים הראשונה:
    a. עדכן את השכן.
12. בצע לולאה על כל שכבת השכנים השנייה:
    a. עדכן את השכן.

כעת, נסביר את הרעיון מאחורי הלוגיקה.

ראשית, אנו מוצאים מי הם השכנים בקרבה ראשונה, ומי הם השכנים בקרבה שנייה, לאחר מכן אנו מבצעים עדכון וקירוב לאותם השכנים באופן שהסברנו לעיל.

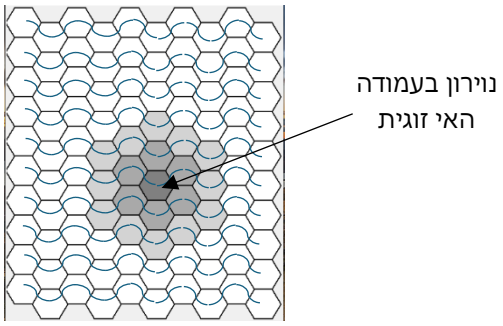
האפקט אותו אנו רוצים ליצור הינו מעין אדוות ברשת של עדכון.

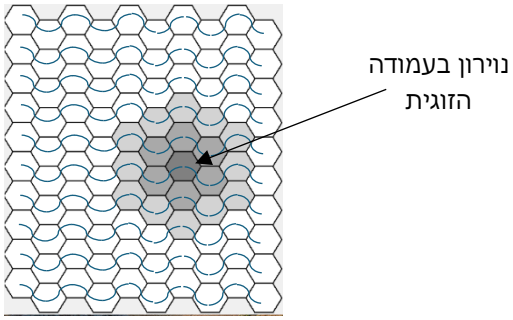


דבר מעניין בעדכון הינו, הדרך בה חושבו האינדקסים של השכנים בכל רמה. בניגוד למימוש רשת בצורת מטריצה פשוטה, כאשר מתבוננים ברשת העשויה ממשושים על חישוב השכנים להיעשות בזהירות.

חישוב שכני הנוירון נעשה לפי מיקום הנוירון ברשת. החישוב משתנה בהתאם לזוגיות העמודה של הנוירון (זוגית או אי זוגית), משום שאנו שומרים בקוד את הנוירונים בצורת מטריצה בעוד שאנחנו מתייחסים לשכני הנוירונים בתצורת תאים משושים.

בשרטוט המוצג הדגשנו כל שורה במטריצה בקו גלי שמבטא ויזואלית את הקשר של מטריצת הנוירונים עם רשת תאי המשושים. מהשרטוט עולה, כי לנוירון הנמצא בעמודה הזוגית ישנם שכנים שונים מהנוירון הנמצא בעמודה האי זוגית במטריצה, וכך הסקנו את מיקומי השכנים ביחס לנוירון המרכז.





חלק ד' – learning rate

ערך ה- learning rate, הינו ערך חשוב ביותר המשפיע על רמת הלמידה בכל שלב בבניית הרשת.

ערך זה מתעדכן במהלך הריצה מאיטרציה לאיטרציה, והינו שונה בין עדכון ה- BMU עצמו, לבין שכניו, ואפילו בין השכבה הראשונה של שכניו לשנייה.

קבענו את ערך הלמידה להיות בתחילת הריצה 0.3, זאת לאחר ניסיונות רבים של ערכים שונים בין 0-1.

רציונל השימוש בערך למידה, הינו לשמר בתהליך הלמידה חלק מהידע הקודם, וכך בעצם ללמוד, הרי עבור ערך למידה גדול מידי נקבל שהרשת תשתנה יתר על המידה בכל איטרציה, ועבור ערך קטן מידי לא תשתנה כמעט בכלל.

את תהליכי העדכון של ה- BMU ושכניו תיארנו בחלקים הקודמים. שם ראינו שבעת העדכון אנו שולחים את ערך הנוירון המתעדכן מהרשת יחד עם ערך למידה מסוים והוקטור אליו אנו מתקרבים לפונקציה מסוימת. כעת, נתבונן בה מקרוב:

```
# Function to calculate the update for a neuron based on a vector and learning rate
def calc_update(vector, neurons, learning_rate):
    return neurons + learning_rate * (vector - neurons)
```

1. החזר את ערך הנוירון ועוד העדכון.

כעת, נסביר את הרעיון מאחורי הנוסחה, עבור N_i הנוירון ה- i, u_j הוקטור ה- j בדאטה סט ו- λ ערך הלמידה עבור האיטרציה ה- t:

$$N_i^{t+1} = N_i^t + \lambda \cdot (u_j - N_i^t)$$

רעיון עדכון הניורון, הינו להפחית את ערך השגיאה בין הנוירון לוקטור, כאשר ערך השגיאה מוגדר כהפרש בין הוקטור לנוירון. בכל עדכון אנו מוסיפים לנוירון את ערך השגיאה עם ערכים מוקטנים על ידי הכפלה בערך הלמידה וכך אנו מקרבים את ערך הנוירון לערכי הוקטור.

לשם דיוק תהליך הלמידה, אנו מורידים מעט את ערך הלמידה בין האיטרציות, כל שככל שנהיה במספר גבוה יותר באיטרציות נשמר יותר מהמידע שכבר למדנו. דבר זה הינו מקובל בתהליכי למידה שונים, כיוון שבשלב מתקדם יותר בלמידה מתרחש יותר דיוק, ואנו רוצים לשמור על הרשת מאוזנת בזמן זה.

עדכון ערך הלמידה מאיטרציה לבאה אחריה, הינה בצורה פשוטה של חילוק ערך הלמידה ב- 2. לאחר ניסיונות רבים גילינו שדווקא העדכון הפשוט הוא שמביא לתוצאות הטובות ביותר, ולכן לבסוף נשארנו אתו.

הסברנו לעיל, כי בעדכון שכבות השכנים נרצה להוריד את עוצמת הלמידה בכל איטרציה, לשם כך אנו מקטינים את ערך הלמידה עבור השכבה הראשונה ע"י חילוק במספר האיטריציות פלוס אחד ועבור השכבה השנייה ע"י חילוק במספר האיטרציות פלוס אחד בריבוע. נעשה זאת על מנת שבתחילת הלמידה הנוירונים השכנים יהיו קרובים בצורתם לנוירון המרכז וישמרו קרבה טופולוגית טובה, ולאחר שהנוירונים יחסית התייצבו ו"מצאו" את מקומם, נרצה לקבע יותר את ערך הנוירון ולשנות פחות את ערך שכניו.

חלק ה' – טיב הפתרון

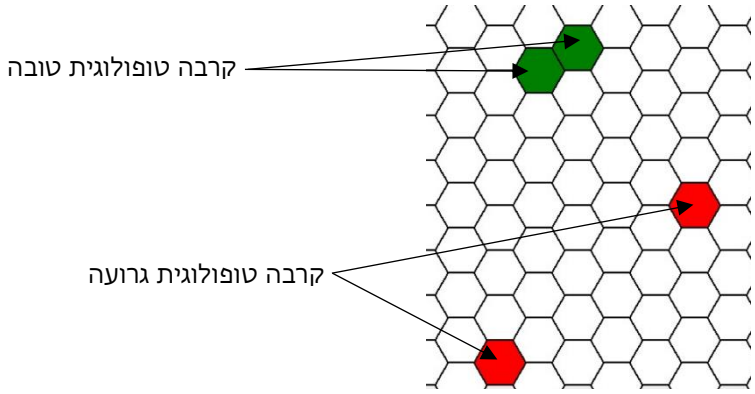
```
# Function to calculate a score
def calc_score(board):
    total_score = 0
    num_sample = 0
    max_euclidean_distance_value = np.sqrt(VECTOR_LENGTH * (255 ** 2))
    max_topological_distance_value = 13

    for vector in DS:
        num_sample = num_sample + 1
        i, j = extract_bmu_idx(vector)
        error_euclidean_distance = 100 * (euclidean_distance(vector, NEURONS_MATRIX[i][j]) / max_euclidean_distance_value)

        first_closest_neuron_idx, second_closest_neuron_idx = find_two_best_neurons(vector)
        topological_distance = board.topological_distance(first_closest_neuron_idx, second_closest_neuron_idx) - 1
        error_topological_distance = 100 * (topological_distance / max_topological_distance_value)

        total_score = total_score + (100 - (0.5 * error_euclidean_distance + 0.5 * error_topological_distance))

    return total_score / num_sample
```

1. נאתחל את הציון ל- 0.
2. נאתחל את מספר הדגימות ל- 0.
3. נגדיר מהו ערך השגיאה האוקלידית הגדול ביותר בין נוירון לוקטור מסוים.
4. נגדיר את המרחק הכי גדול בין שני נוירונים בלוח המשושים של רשת הנוירונים.
5. בצע לולאה על כל הוקטורים מהדאטה סט:
    a. נוסיף למספר הדגימות הקיימות 1.
    b. נחלץ את מיקום ה- BMU הקרוב ביותר לדגימה.
    c. נחשב את המרחק בין הדגימה ל- BMU, נחלק בערך המקסימלי לשגיאה ונכפיל ב- 100.
    d. נמצא את מיקום שני הנוירונים הכי קרובים לדגימה.
    e. נחשב את המרחק הטופולוגי בין שני הנוירונים שמצאנו ונפחית 1.
    f. נחשב את שגיאת המרחק הטופולוגי ע"י חילוק המרחק הטופולוגי מינוס אחד במרחק הכי גדול שבין שני נוירונים בלוח ונכפיל ב– 100.
    g. נוסיף לציון הכולל 100 ונחסר את החיבור בין חצי המרחק האוקלידי וחצי מהמרחק הטופולוגי.
6. נחזיר את הממוצע של כל ציוני הדגימות.

נסביר שני מוטיבים עליהם מושתת טיב הפתרון.

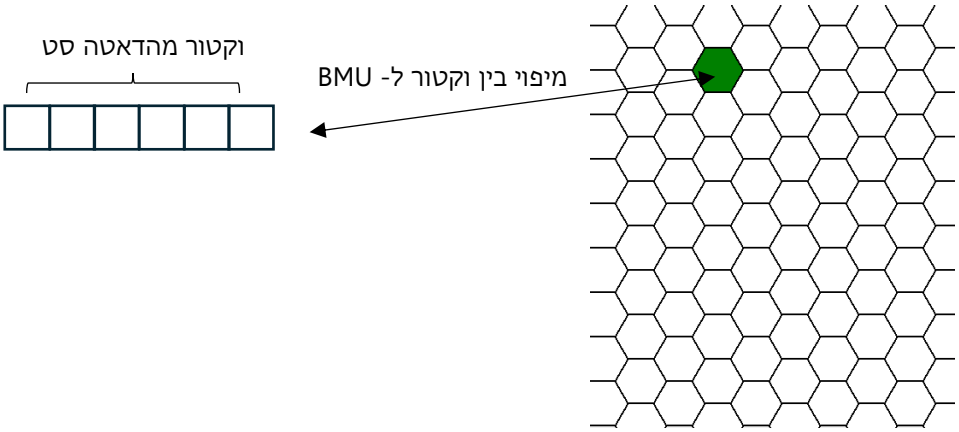
1) קרבה טופולוגית ברשת –

אנו מודדים מידת הקרבה בין שני נוירונים ברשת על ידי חישוב מספר המשושים המינימלים המפרידים בין שני הנוירונים.



2) מרחק הטעות –

אנו מודדים לכל וקטור את המרחק האוקלידי בינו ל- BMU שלו.



כעת, נסביר את הרעיון מאחורי חישוב הציון.

לכל וקטור אנו מחשבים את ערך שגיאת המרחק האוקלידי בינו לבין ה- BMU שלו ומנרמלים את הציון לטווח ערכים בין 0 ל- 100. אנו עושים זאת באמצעות חילוק המרחק האוקלידי שקיבלנו במרחק האוקלידי המקסימלי והכפלת התוצאה ב-100. בנוסף, לכל וקטור נחשב את המרחק הטופולוגי בין שני הנוירונים שהכי קרובים לוקטור ונחסיר אחד מהמרחק, זאת משום שהמרחק הטוב ביותר שנוכל לקבל בין שני נוירונים הוא אחד, כאשר הניורונים שכנים. בכדי לנרמל את המרחק הטופולוגי נחלק אותו במרחק הגדול ביותר שבין שני נוירונים ברשת ונכפיל ב-100. לבסוף, נחבר את שגיאת המרחק האוקלידי ושגיאת המרחק הטופולוגי וננרמל ל- 100, כך שאנו נותנים את אותו המשקל לשני שגיאות המרחקים ואת התוצאה נחסר מ- 100. כך נקבל שהציון של כל וקטור הוא קטן יותר ככל שמרחקי השגיאה גדולים יותר.

בסוף, לאחר חיבור כל ציוני הוקטורים נחשב להם ממוצע ונקבל את הציון אותו רצינו.

נסמן ב- N_1^j, N_2^j את הנוירונים הקרובים ביותר (ה- BMU וסגנו בהתאמה) לוקטור u_j הוקטור ה- j בדאטה סט. נגדיר $TD(N_i, N_j)$ להיות המרחק הטופולוגי בין שני נוירונים ברשת המשושים, ואת $ED(N_i, u_j)$ להיות המרחק האוקלידי בין נוירון לוקטור.

סה"כ נקבל את הנוסחה הבאה:

$$SCORE = \frac{\sum_{j=1}^{10000} 100 - \left(\left(100 \cdot \frac{TD(N_1^j, N_2^j)}{\max_{k,l}\{TD(N_k, N_l)\}} \right) \cdot 0.5 + \left(100 \cdot \frac{ED(N_1^j, u_j)}{\sqrt{784 \cdot 255^2}} \right) \cdot 0.5 \right)}{10000}$$

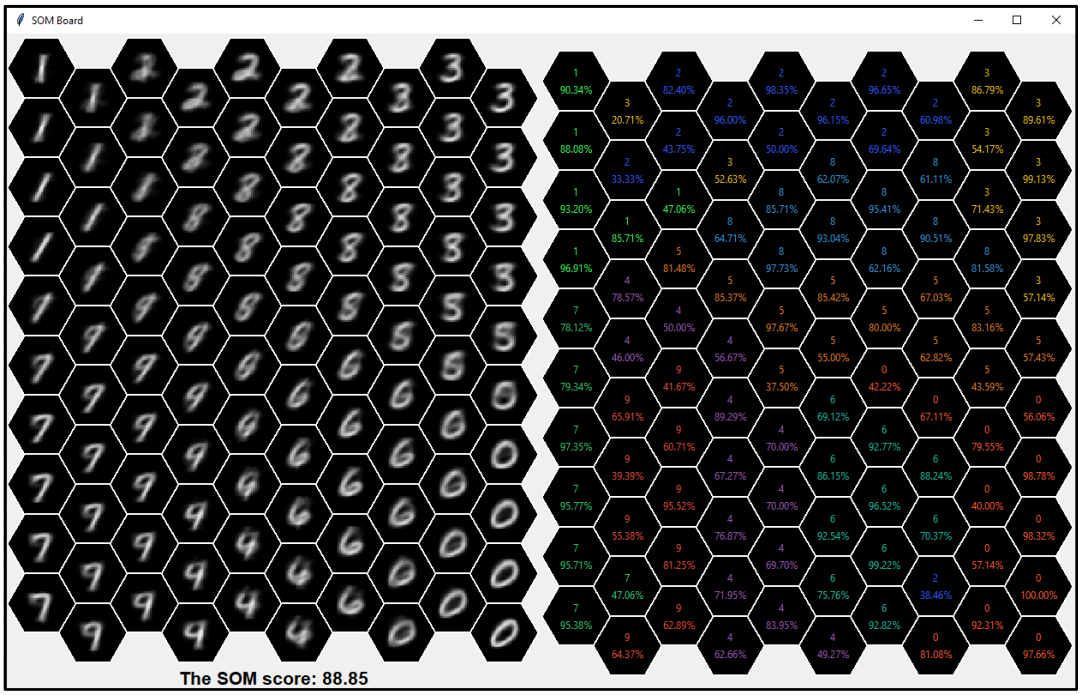
חלק ו' – דוגמות הרצה

תחמנו את זמן הלמידה למינימום בין 60 שניות (לפני ביצוע האיטרציה האחרונה) ל- 5 איטרציות למידה, כיוון שראינו שכך המודל לומד בצורה הטובה ביותר ביחס בין זמן לדיוק.

הצגת הפתרון ותוצאות הריצה מתבצעת על ידי הדפסת שני לוחות משושים:

1) לוח משושים המציג את תמונתם של וקטורי הנוירונים בסיום הלמידה.
2) לוח משושים בו כל נוירון מציג את זהות הספרה הדומיננטית שהוא מייצג ואת אחוז ההופעה שלה, כאשר כל ספרה מודפסת בצבע שונה בכדי לאפשר ויזואליות טובה יותר.

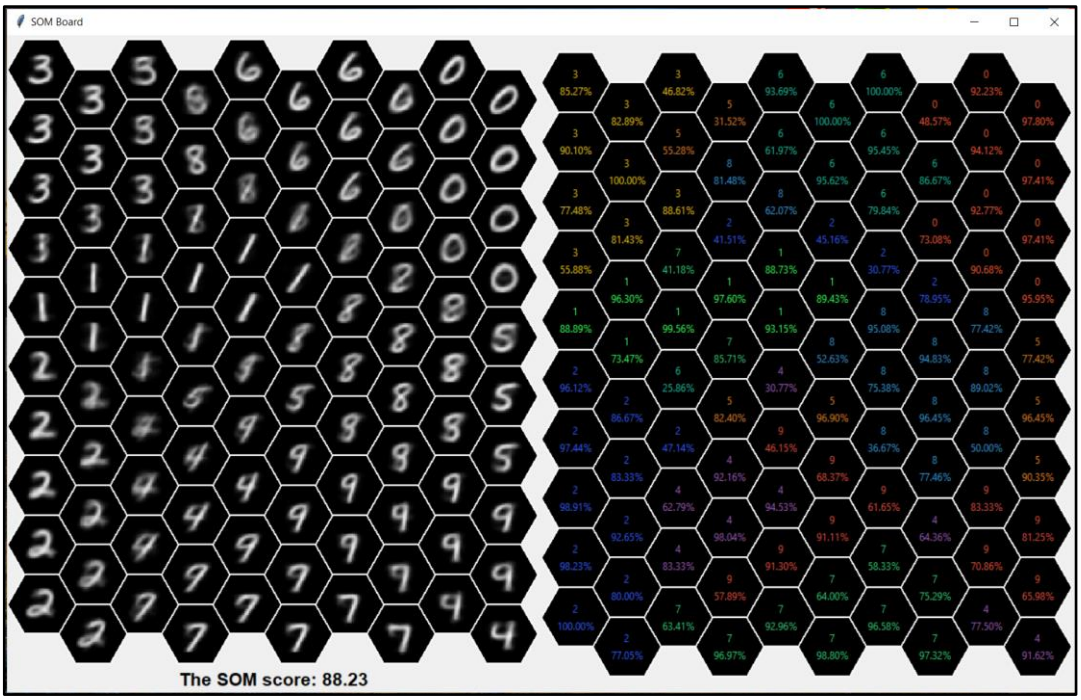
כעת, נציג מספר דוגמות הרצה. נציין כי לכל דוגמת הרצה הצגנו את הציון שלה, כאשר הציון מתחלק בין 0 ל-100 (כשם שביארנו בחלק ה'), נזכיר שככל שציון גדול מציין תוצאת למידה טובה יותר.



הרצה ראשונה

ציון: 88.85 | זמן הלמידה: כ- 41.48 שניות | ערך למידה התחלתי: 0.5 | ערך למידה שכבת שכנים ראשונה: ערך למידה חלקי 2 | ערך למידה שכבת שכנים שניה: ערך למידה חלקי 4 | הפחתת ערך למידה: ערך הלמידה חלקי 2

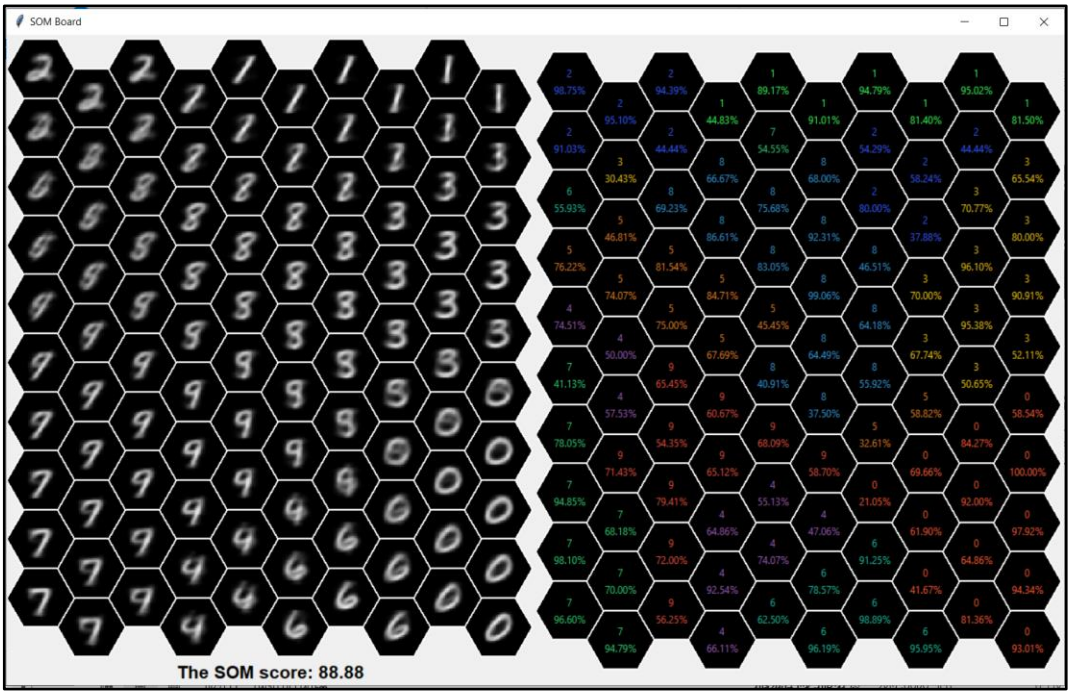
קיבלנו ציון גבוה עם קבוצות מלוכדות של מספרים זהים שקרובים למספרים, שויזואלית נראים דומים להם, אך תמונות וקטורי הנוירונים מטושטשים.



הרצה שניה

ציון: 88.23 | זמן הלמידה: כ- 42.66 שניות | ערך למידה התחלתי: 0.5 | ערך למידה שכבת שכנים ראשונה: ערך למידה חלקי מספר האיטרציה פלוס 1 | ערך למידה שכבת שכנים שניה: ערך למידה חלקי מספר האיטרציה פלוס 1 בריבוע | הפחתת ערך למידה: ערך הלמידה חלקי 2

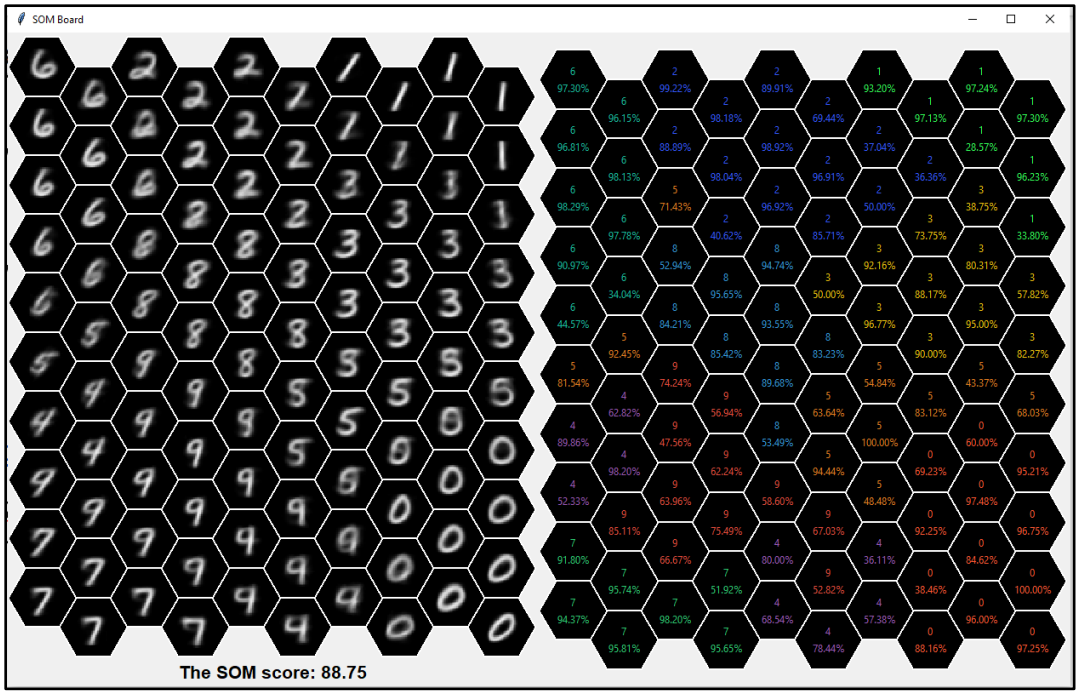
הציון נמוך יותר, אך ניתן לראות את המספרים בבירור, המספרים מתקבצים לקבוצות, אך יש קצת פיזור בין הקבוצות.



הרצה שלישית

ציון: 88.88 | זמן הלמידה: כ- 40.38 שניות | ערך למידה התחלתי: 0.5 | ערך למידה שכבת שכנים ראשונה: ערך למידה חלקי 2 | ערך למידה שכבת שכנים שניה: ערך למידה חלקי 4 | הפחתת ערך למידה: ערך הלמידה חלקי 2

כאן קיבלנו את הציון הכי גבוה עד כה, יחד עם קבוצות מלוכדות של מספרים זהים הקרובים למספרים שויזואלית נראים דומים להם, אך תמונות וקטורי הנוירונים מטושטשים.



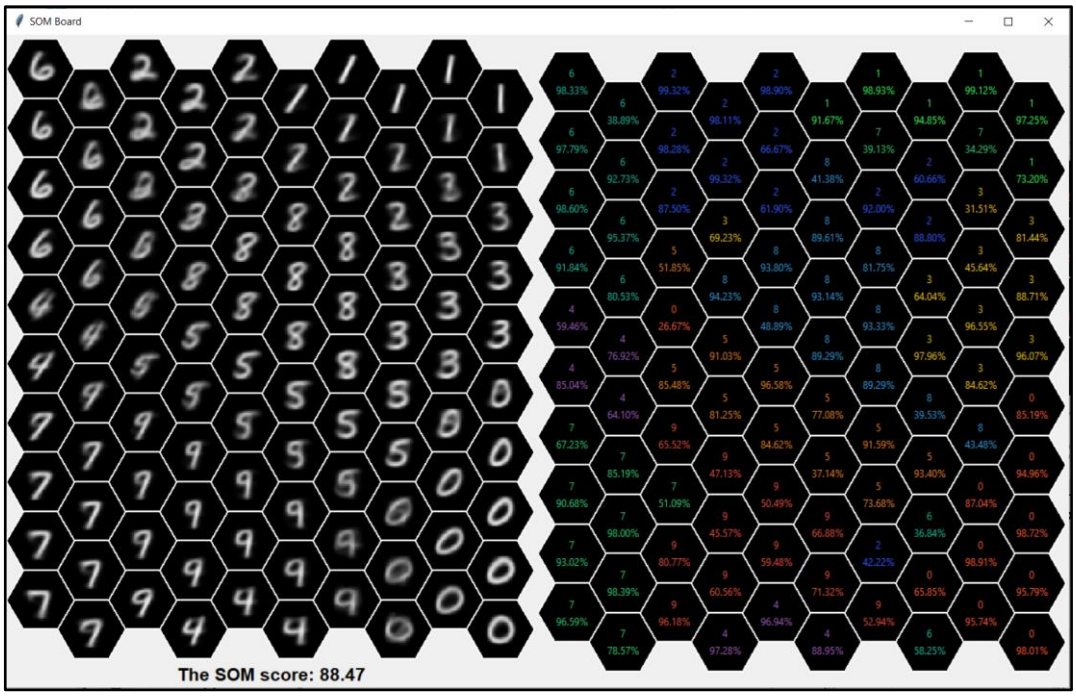
הרצה רביעית

ציון: 88.75 | זמן הלמידה: 40.47 שניות | ערך למידה התחלתי: 0.3 | ערך למידה שכבת שכנים ראשונה: ערך למידה חלקי מספר האיטרציה פלוס 1 | ערך למידה שכבת שכנים שניה: ערך למידה חלקי מספר האיטרציה פלוס 1 בריבוע | הפחתת ערך למידה: ערך הלמידה חלקי 2

קיבלנו ציון גבוה עם קבוצות מלוכדות של מספרים זהים שקרובים למספרים שויזואלית נראים דומים להם למרות שהציון קצת יותר קטן מההרצה הקודמת, אך תמונות וקטורי הנוירונים ברורים.

הרצה חמישית

ציון: 88.47 | זמן הלמידה: 40.48 שניות | ערך למידה התחלתי: 0.3 | ערך למידה שכבת שכנים ראשונה: ערך למידה חלקי מספר האיטרציה פלוס 1 | ערך למידה שכבת שכנים שניה: ערך למידה חלקי מספר האיטרציה פלוס 1 בריבוע | הפחתת ערך למידה: ערך הלמידה חלקי 3



קיבלנו ציון פחות טוב מההרצה הקודמת עם קצת יותר פיזור של המספרים וגם קל לראות שהתוצאה פחות טובה מההרצה הקודמת.

לסיכום, ההרצה הכי טובה היתה ההרצה הרביעית, כיוון שתמונות וקטורי הנוירונים הוצגו בברור עם הציון השני הכי גבוה מבין כל ההרצות - ציון 88.75. אומנם ההרצה עם הציון הכי גבוה הייתה ההרצה השלישית עם ציון 88.88, אך תמונות וקטורי הנוירונים לא היו ברורות וההפרש בין ציוני הריצות לא היה גבוה, לכן, סה"כ את האיזון המוצלח יותר קיבלנו בהרצה הרביעית ועל כן אלו הפרמטרים עימם נשארנו.

חלק ז' – הוראות הרצה

על מנת להריץ את הקוד, יש להוריד את הקובץ **main.exe**, מהקישור הבא –

https://drive.google.com/drive/folders/1_cArBImj4RPoiB5zf3yOcLt81DkENXAI?usp=sharing

יש לשים קבצי csv אחד עם דאטה עבור הוקטורים המלמדים בשם – "digits_keys.csv", והשני עם הלייבלים של הוקטורים בשם - "digits_test.csv", בתיקייה יחד עם קובץ ההרצה "main.exe", וללחוץ עליו פעמיים.