אביטל רוז, ת.ז 318413408, מדעי המחשב,
רננה דרומר, ת.ז 316332238, ביולוגיה חישובית

הרצת הקובץ :

לחיצה כפולה על קובץ ה-py שהוגש.

נציין שהשתמשנו בחבילות הבאות :

Random, matplotlib, numpy, math, sklearn.neighbors

פלט ההרצה : גרף SOM ופלט עבור כל תא ברשת.

ביולוגיה חישובית- תרגיל 3

תיעוד קוד :

מטריצת som : על מנת להקל על החישובים, יצרנו מטריצה אשר כל שורה בה תואמת לוקטור משושה יחיד (כלומר, אינדקס השורה תואם לאינדקס המשושה). וקטורי המשושה אותחלו כך שכל שדות הוקטור הם בין 0 ל-0.2, כיוון שראינו בבדיקה ידנית שערכי וקטורי הערים תחומים בערך בטווח זה, לאחר הנרמול.

מחלקת Hexagon : (מייצגת כל אחד מהמשושים על פני ה-grid)

שדות המחלקה :

שדה	שם בקוד	תפקיד
מרכז	self. center	נועד לשמירת מיקום מרכז המשושה על פני ה-grid
גודל	self.size	אורך צלע המשושה
פינות	self.corners	מייצג את מיקום קודקודי המשושה
צלעות	self.edges	מייצג את צלעות המשושה
וקטור	self.vector	מאותחל כ-None, מקבל שורה ממטריצת המשושים, בהמשך יעודכן לפי וקטורי הערים.
ערים	self.cities	מאותחל כרשימה ריקה, ויעודכן בהמשך הריצה. מייצג את אוסף הערים ה-"שייכות" לאותו משושה ב-grid.
שכנים	self.neighbours	מאותחל כמילון ריק. בהמשך הריצה יעודכן לפי הקוד. כל משושה מקבל משכניו הראשונים את שכניהם.
ממוצע המצב הסוציו-אקונומי	self.average_social_economic_state	מאותחל כ-None. בהמשך הריצה יעודכן כממוצע הסוציו-אקונומי של הערים שישובצו לאותו משושה.

במחלקת hexagon מאותחלים השדות שנכתבו לעיל.

תפקיד המתודות במחלקה כולל הן יצירת משושים שכנים (כלומר יצירת הלוח), חישוב פרטי המשושים (מיקום הקודקודים והקשתות, חישוב נקודת המרכז), והן חישוב ממוצע סוציו-אקונומי של הערים ששובצו למשושה, עדכון הוקטור המייצג את המשושה. כמו כן, ישנן מתודות למציאת מרחק בין שכנים וכן מציאת "טבעת השכנים" הראשונה, השניה והשלישית של משושה, כאשר הכוונה ב-"טבעת" שכנים היא השכנים שמסביב לאותו משושה, במעגל הראשון שסביבו, השני והשלישי.

מתודה חשובה במחלקה זו הינה מתודת עדכון הוקטור לפי כלל העדכון. תיאור נוסחת כלל העדכון כפי שנלמדה בהרצאה :

The update rule:

$$N_k(t+1) = N_k(t) + \underbrace{\alpha(t)}_{\text{Learning constant}} \cdot \underbrace{h_{ki}(t)}_{\text{Neighborhood}} \cdot \underbrace{[V_i - N_k(t)]}_{\text{Current error}}$$

מימוש נוסחת עדכון וקטור, נעשה בעזרת חישוב על ידי הפרמטרים הבאים :

1. FIRST_EFFECT : מסמל את ה-learning-rate, נקבע לאחר מספר ניסויים ל-0.4.
2. Ring : מסמלת את ה-neighborhood של המשושה. ככל שה-ring גדול יותר, המשושה רחוק יותר מהמשושה המרכזי, ולכן האפקט עליו יהיה קטן יותר. למשל, עבור ring_2, האפקט יחושב : 2=2-4.

כך שאת הנוסחה מימשנו באופן הבא :

self.vector = self.vector + FIRST_EFFECT * (4 - ring) * (vector - self.vector)

מחלקת Line : המחלקה מקבלת שתי נקודות, והקו המיוצג הוא הקו שנמתח ביניהן. מטרת יצירת הקו היא לשם יצירת המשושים על פני הלוח.

שדות המחלקה :

שדה	שם בקוד	תפקיד
נקודה א	self.point_a	מאותחל כנקודה שהתקבלה. נקודה בקצה אחד של הקו.
נקודה ב	self.point_b	מאותחל כנקודה שהתקבלה. נקודה בקצה השני של הקו.

תפקיד המתודות במחלקה כולל מציאת המרחק בין הנקודות, חישוב משוואת קו ישר, והדפסת שורה.

מחלקת Point : המחלקה מקבלת שתי קורדינטות (x,y), והן יאפשרו ליצור את הלוח בצורה נוחה.

שדות המחלקה :

אביטל רוז, ת.ז 318413408, מדעי המחשב,
רננה דרומר, ת.ז 316332238, ביולוגיה חישובית

שדה	שם בקוד	תפקיד
קורדינטה א	self.x	מאותחל כ-x שהתקבל.
Eurshbyv c	Self.y	מאותחל כ-y שהתקבל.

תפקיד המתודות במחלקה כולל מציאת מרחק לנקודה אחרת והדפסת נקודה.

<u>מהלך קוד כללי</u>:

תחילה, קראנו את שורות קובץ ההצבעות אל תוך מילון הצבעות, כאשר כל שורת הצבעות מתורגמת לוקטור באורך כמות המפלגות.

נרמול הוקטורים – ניסינו מספר שיטות נרמול שונות (min-max, z-score, percentage). תיאור השפעות הנרמול השונות מוצג בתוצאות.

יצירת הלוח – ראשית, יצרנו תבנית של משושה קטן, שיהווה התבנית הקטנה ביותר, כאשר לכל משושה ישנו וקטור אקראי בגודל כמות המפלגות. שנית, ציירנו את המשושים בזה אחר זה, מנקודה שמאל-עליונה, עד סוף השורה הימנית, כך שכל משושה מצייר את שכנו הימני. לאחר מכן חזרנו למשושה הראשון, וציירנו את המשושה שמתחתיו, ולאחר מכן זה המשיך לצייר רקורסיבית את שכניו בהמשך השורה לכיוון EAST. באופן זה ציירנו את כל שורות הלוח.

מציאת שכנים – יצרנו פונקציה (find_neighbours) שתאפשר לכל משושה למצוא את שכניו, במספר "טבעות" סביבו. קלט הפונקציה הינו המשושה המרכזי, ופלט הפונקציה הוא סט המשושים שמקיפים אותו. בטבעת הראשונה (הסמוכה ביותר למשושה המרכזי), הפלט הינו המשושים אשר באים איתו במגע סביבו. בטבעת השניה המורכבת למעשה משכני כל אחד משכני המשושה המרכזי בדרגה הראשונה, בדקנו את השכנויות לכל אחד ממשושי הטבעת הראשונה, ולשם הימנעות כפילויות של משושים, שמרנו את המשושים ב-set. באופן דומה יצרנו את הטבעת השלישית.

Train- עברנו בלולאה על פני כל אחת מהערים במילון ההצבעות, ובאמצעות מרחק KNearestNeighbors (חישוב שכנים לפי מרחק אוקלידי) מצאנו את שני המשושים שהוקטור שלהם הכי קרוב לוקטור ההצבעות של אותה עיר. לאחר מכן, קירבנו את וקטור המשושה הנבחר לוקטור של העיר, וקירבנו את וקטורי המשושים בטבעות השכנים בכמות קטנה יותר ככל שמתרחקים (טופוגרפית) מהמשושה הנבחר.

חישוב ה-loss- יצרנו שתי פונקציות loss. loss a: עוברת על פני כל המשושים, וסוכמת את ה-loss בין כל וקטורי הערים שמשויכות למשושה והוקטור של המשושה, זאת באמצעות שגיאת RMS. למעשה, פונקציית ה-loss הראשונה מכמתת את ההומוגניות בתוך משושה, כלומר היכולת של האלגוריתם לשבץ ערים דומות לאותו משושה. loss b: פונקציית ה-loss השנייה עוברת על פני כל הערים ומחשבת את המרחק הטופולוגי בין המשושה שנבחר לאותה עיר (ההתאמה הטובה ביותר) לבין ההתאמה השנייה הטובה ביותר. למעשה, פונקציית ה-loss השנייה מכמתת את יכולת הקיבוץ המרחבי של ערים דומות במשושים קרובים. לאחר מכן, לשם שקלול תוצאות ה-loss שהתקבלו משתי הפונקציות, שקללנו את שני ה-losses שהתקבלו כך שלראשון ניתן משקל גבוה יותר (0.7) ולשני משקל נמוך יותר (0.3), כיוון שהשני קיבל תוצאות גדולות בסדר גודל מה-loss הראשון.

<u>מענה לשאלות</u>:

חישוב המרחק בין ישוב בקלט לבין התא ב-grid שמייצג אותו: כפי שכתוב בחלק ה-train תחת כותרת "מהלך קוד כללי", החישוב נעשה על פי מרחק אוקלידי (KNN).

קירוב התא אליו מופה קלט מסוים לאותו קלט ואופן שינוי שכניו: החישוב מפורט תחת מחלקת hexagon.

השפעת סדר הצגת היישובים למערכת: לשם בדיקת נושא זה, בדקנו אפשרויות שונות להצגת היישובים בסדרים שונים - לפי הסדר המצוין בקובץ ה-csv, לפי shuffle בין הערים, לפי מצבן הסוציו-אקונומי ולפי מספר המצביעים בעיר. מסקנתנו היתה שהסדר הטוב ביותר הינו מספר הקולות שהצביעו בעיר, שכן הוביל ל-loss נמוך וכן להצגת clustering בצורה מיטבית. מפורט בחלק התוצאות ומסקנות.

בחירת הפתרון להגשה מבין הפתרונות השונים: כפי שמוצג בחלק התוצאות ומסקנות, בחרנו זאת על פי יצירת פונקציות loss ובחירת הפתרון בעל ה-loss הנמוך ביותר.

<u>תצוגה</u>:

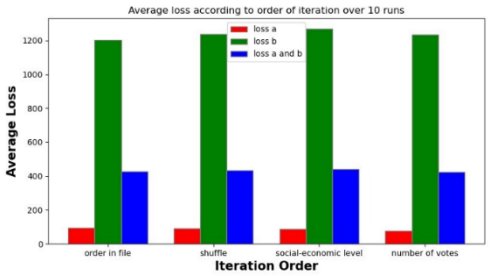
כדי ליצור תצוגה מתאימה, השתמשנו בחבילת matplotlib. באמצעותה יצרנו צבע מתאים לכל משושה על פי רמתו הסוציו-אקונומית הממוצעת.

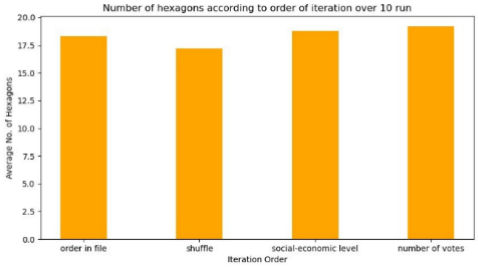
<u>תוצאות ומסקנות</u>:

<u>השפעת סדר הכנסה</u>: לשם בדיקת השפעת הכנסת סדר הערים, בדקנו את הכנסתן לפי סדרים שונים: לפי הסדר המצוין בקובץ ה-csv, לפי shuffle בין הערים, לפי מצבן הסוציו-אקונומי ולפי מספר המצביעים בעיר. בגרף הימני מתוארת השפעת סדר ההכנסה על ממוצע הloss של 10 ריצות (loss מסוג A באדום, מסוג B בירוק, בכחול שקלול של שניהם). הממוצע דומה בין כל סדרי ההכנסה, אך לאחר הדפסת הערכים התקבל כי בכל המדדים עמודת סדר ההכנסה לפי כמות המצביעים היא הנמוכה ביותר, כלומר בעלת ה- loss הקטן ביותר. בגרף השמאלי ניתן לראות את ממוצע מספר

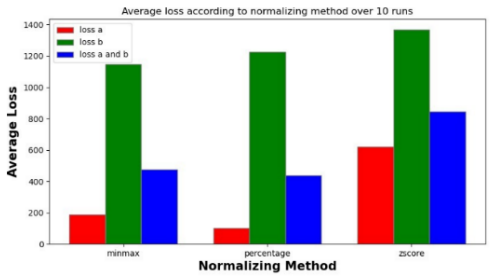
אביטל רוז, ת.ז 318413408, מדעי המחשב,
רננה דרומר, ת.ז 316332238, ביולוגיה חישובית

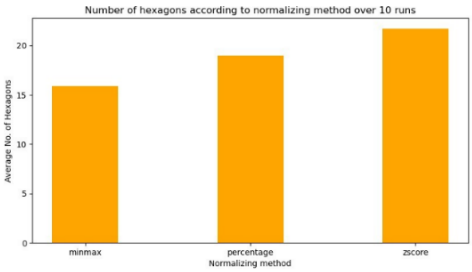
המשושים שמשובצים בהם ערים לפי סדר ההכנסה. ניתן לראות שעבור סדר הכנסה לפי מספר התושבים מתקבלים הכי הרבה משושים בממוצע. ככל שיש יותר משושים, התוצאה יפה יותר לעין וגם מראה על פיזור מרחבי גדול יותר שיכול להוריד את השגיאה (כיוון שיש פחות ערים בכל משושה, יש פחות וקטורי ערים שמשפיעים על וקטור המשושה, ולכן הוא יכול להיות מדויק יותר עבור הערים שכן נמצאים בו). התוצאה השנייה המוצלחת הינה כסידור לפי מצב סוציו אקונומי, ולאחר מכן הצלחה דומה עבור shuffle וסדר הכנסה לפי הקובץ.





<u>השפעת שיטות נרמול שונות</u>: לשם בדיקת השפעת נרמול הדאטא על התוצאות, הרצנו 10 ריצות עבור שלוש שיטות נרמול שונות – minmax, percentage, z-score. כפי שניתן לראות, קיימת השפעה לשיטת הנרמול על תוצאות הריצה. בגרף ימין מוצגת ההשפעה הממוצעת על הloss (סוג A באדום, סוג B בירוק ושקלול בכחול) בגרף שמאל מוצגת ההשפעה על כמות המשושים הממוצעת. כפי שניתן לראות בגרף השמאלי, מספר המשושים הגדול ביותר מתקבל בשיטת z-score, ולאחר מכן בסדר יורד בשיטת percentage ובשיטת minmax. עם זאת, כפי שניתן לראות בגרף הימני, ממוצע ה-loss הקטן ביותר מתקבל עבור שיטת minmax, לאחר מכן בסדר עולה על percentage ובעקבותיו על z-score. מכיוון שלהבנתנו, הצורך ב-loss קטן יותר היה משמעותי מאשר מספר המשושים המתקבלים, בחרנו בשיטת נרמול percentage, יחד עם מספר ביניים ממוצע של המשושים.

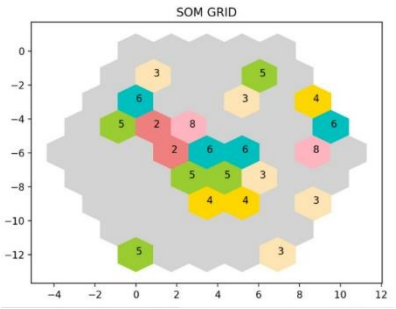




<u>סיכום</u>: לסיכום, יצרנו grid של משושים, בתחילה כל אחד מהם ייוצג על ידי וקטור בעל ערכים רנדומיים. כדי לייצג כל אחת מהערים, יצרנו וקטור תואם לכל עיר. לשם יצירת ה-clusters, ערכנו בדיקת מרחקים בין וקטורי הערים לוקטורי ה-grid, ובאופן כזה הערים "שובצו" למשושים ב-grid, ווקטור כל משושה שונה בהתאם. תוצאות הניסוי הראו כי אכן התבצע clustering כנדרש, שכן ערים בעלות אוכלוסייה דומה שובצו לאותם אזורים ברשת.

בחרנו בנרמול לפי percentage שכן מנרמול זה התקבל ה-loss הנמוך ביותר על פני כל המדדים. שקלנו להשתמש בנרמול z-score משום שבו התקבל מספר משושים גדול יותר (המציג תוצאה יפה יותר לעין), אך בחרנו שלא להשתמש בו כיוון שקיבל loss גבוה בהרבה מבין כל שיטות הנירמול, במיוחד ב-loss מסוג a, המתאר את התכנסות המערכת, ולהבנתנו משמעות ה-loss משמעותית יותר ממספר המשושים על פני הלוח. בנוסף ראינו שעבור סדר הכנסה לפי כמות המצביעים קיבלנו את התוצאות הטובות ביותר.

מצורפת תמונת הרשת הנבחרת מתוך 10 האיטרציות להרצת הקוד:



בגרף ניתן לראות את הממוצע הסוציו-אקונומי של הערים שהשתבצו לכל אחד מהמשושים, כל צבע מייצג מצב ממוצע שונה, אשר כתוב במרכזו. ניתן לראות שבוצע clustering בצורה יחסית טובה, כלומר צבעים דומים התקבצו יחסית לאותו אזור. יצוין כי בבדיקה שלנו, ראינו כי ההתקבצות אכן נעשתה באופן תואם לדמיון אוכלוסייה בין היישובים השונים. למשל, יישובים מהמגזר הערבי התקבצו יחד, יישובים מהמגזר החרדי, וכן יישובים מהמגזר הכללי. פירוט ערים לדוגמא מצורף בנספחים.

אביטל רוז, ת.ז 318413408, מדעי המחשב,
רננה דרומר, ת.ז 316332238, ביולוגיה חישובית

נספחים:

תוצאות בדיקות הנורמליזציה על פי שיטות נירמול שונות (minmax, percentage, z-score)

minmax: average a: 186.6590113893425 average b: 1145.8130540781287 average a and b: 474.40522419597846 average hexagons: 15.9

percentage: average a: 102.2547162592379 average b: 1225.1409216149914 average a and b: 439.12057786596387 average hexagons: 19.0

zscore: average a: 619.7484262440177 average b: 1367.4534972830693 average a and b: 844.0599475557331 average hexagons: 21.7

תוצאות בדיקת השפעת סדר הכנסת הערים, על פי סדרים שונים (הסדר כפי שבקובץ המקורי, שאפל, מצב סוציו-אקונומי, מספר הקולות)

order in file: average a: 94.69205401748162 average b: 1204.739656268332 average a and b: 427.70633469273673 average hexagons: 18.3

shuffle: average a: 89.03846835889338 average b: 1236.6606361952304 average a and b: 433.32511870979454 average hexagons: 17.2

social-economic level: average a: 85.45985207208172 average b: 1268.2865346264196 average a and b: 440.30785683838315 average hexagons: 18.8

number of votes: average a: 77.51955261512322 average b: 1234.5860610916475 average a and b: 424.63950515808045 average hexagons: 19.2

פירוט הערים ששובצו לכל משושה:

0 ['Bethel']

1 []

2 []

3 []

4 []

5 []

6 ['Baana', 'Gat', 'Daboria', 'Deer Alasad', 'Turan', 'Tira', 'Tamra', 'Kabul', 'Kafr Qasim', 'Nahaf', 'Ein Mahel', 'Araba', 'Sheb']

7 []

8 ['Kfar Bara']

9 ['Bnei Brak', 'Rechasim']

10 []

11 []

12 []

13 []

14 []

15 []

16 []

17 []

18 []

19 []

20 ['Even Yehuda', 'Binyamina Givat Ada', 'Givatayim', 'Har Adar', 'Zichron Yaakov', 'Yesod Hammala', 'Kochav Yair', 'Kfar Vradim', 'Kfar Kama', 'Kfar Shmaryahu', 'Lehavim', 'Mevaseret Zion', 'Metula', 'Savyon', 'Omer', 'Pardes Hana', 'Kazir', 'Kiryat Ono', "Kiryat Tiv'on", 'Rosh Pina', 'Ramat Gan', 'Ramat Hasharon', 'Raanana', 'Tel Aviv Jaffa']

21 []

22 ['Ofakim', 'Or Yehuda', 'Or Akiva', 'Oranit', 'Azur', 'Eilat', 'Elyakhin', 'Ariel', 'Beer Sheva', 'Beit Arye', 'Beit Dagan', 'Beit Shean', 'Givat Shmuel', 'Gedera', 'Gan Yavne', 'Hadera', 'Holon', 'Hatzor HaGlilit', 'Harish', 'Tiberias', 'Tirat Carmel', 'Yavneel', 'Yavne', 'Yeruham', 'Lod', 'Migdal', 'Meitar', "Ma'ale Adumim", 'Tarshiha', 'Nahariya', 'Netivot', 'Netanya', 'Sagur', 'Egar', 'Acre', 'Afula', 'Petah Tiqwa', 'Kiryat Ata', 'Kiryat Gat', 'Kiryat Malachi', 'Kiryat Ekron', 'Kiryat Shmona', 'Rama', 'Rosh HaAyin', 'Ramla', 'Sderot', 'Shlomi']

23 []

24 []

25 []

26 ['Bueeninogidat', 'Bir al-Maksur', 'Beit Shemesh', 'Basmat Tivon', 'Givat Zeev', 'Galgulia', 'Zarzir', 'Hura', 'TubaZangaria', 'Kassifa', 'Kaabiya Tbashhagagra', 'Kfar Kana', 'Kfar Manda ', 'Lakia', 'Mashhad', 'Sakhnin', 'Ilut', 'Rahat', 'Shibli Um Elganam', 'Segev Shalom', 'Tel Sheva']

27 ['Beit Gan', 'Hurfish', 'Osfia', "Peki'in Buqiya"]

28 []

29 ['Ashdod', 'Jerusalem', "Modi'in Illit", 'Arad', "Kiryat Ye'arim"]

30 []

31 ['Emanuel']

32 []

33 []

34 ['Nof Hgalil ']

35 []

36 []

37 ['Magar']

38 []

39 []

40 []

41 []

42 []

43 []

44 []

45 []

46 ['Alfei Menashe', 'Beer Yaakov', 'Ganei Tikva', 'Hod Hasharon', 'Herzliya', 'Haifa', 'Yehud Monoson', 'Yokneam Illith', 'Kfar Yona', 'Kfar Saba', 'Carmiel', "Modi'in Maccabim-Reut", 'Mazkeret Batya', 'Ness Ziona', 'Nesher', 'Pardesiya', 'Kadima Tzoran', 'Kiryat Bialik', 'Kiryat Motzkin', 'Rishon Lezion', 'Rehovot', 'Ramat Yishai', 'Shoham', 'Tel Mond']

47 ['Ashkelon', 'Benei Aish', 'Bat Yam', 'Guls', 'Dimona', 'Yanuch Gat', 'Yarka', 'Kasrasmie', 'Migdal Haemek', 'Katzrin', 'Kiryat Yam']

48 []

49 []

50 ['Maale Ephraim', 'Mitzpe Ramon', 'Kdomim', 'Kiryat Arba', 'Karni Shomron']

51 []

52 []

53 ['Elkana', 'Efrat', 'Zefat']

54 ['Abu Gosh', 'Abu Sanan', 'Umm al-Fahm', 'Ixal', 'Ibelin', 'Baka Algarbia', 'Basma', 'Gadidamacher', 'Gesser Azarka', 'Gush Halav', 'Deer Hanna', 'Zemer', 'Taibeh', 'Yafia', 'Kfar Yassif', 'Kfar Kara', 'Majd al-Krum', 'Mazraa', 'Miilya', "Ma'ale Eron", 'Nazareth', 'Eilabun', 'Arara', 'Fureidis', 'Fasuta', 'Kalenswa', 'Reina', "Shefar'am"]

55 ["Dali'at el Carmel"]

56 []

57 ['Elad', 'Beitar Illit']

58 ['Arara in the Negev']

59 []

60 []