מגישים: רביב חכם, 208387951
פלג חכם, 208387969

ביולוגיה חישובית – תרגיל 3

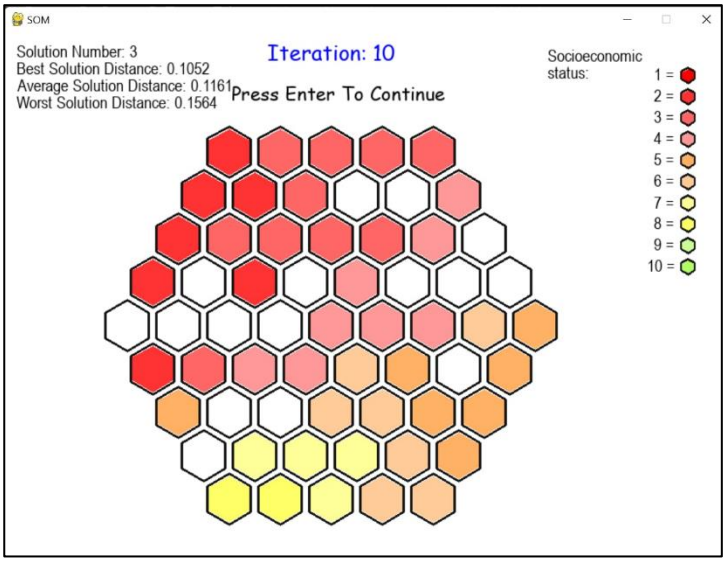
קישור לגיט:

(בגיט מופיעים כל קבצי הפיתון המרכיבים את התוכנית שבנינו, הדוח הנ"ל, ובפרט קובץ ההרצה main.exe)
https://github.com/peleg5050/Biologi3/tree/master

הנחיות והסבר הרצה:

ישנן שתי דרכים להריץ את התוכנית שבנינו:
דרך ראשונה (הקלה): יש ליצור ולהיכנס לתיקייה בה מופיע קובץ ההרצה main.exe והקובץ Elec_24.csv (אשר סופק לנו), יש ללחוץ (לחיצה כפולה) על הקובץ: main.exe, לאחר ההרצה יפתח חלון שיציג את 10 הפתרונות של לוח ההקסגון, ולבסוף יוצג גרף שמתאר את המרחק הממוצע (מפורט בהמשך) ורשימת הערים עם המיפויים תופיע בדף הטרמינל אשר יפתח באופן אוטומטי.
דרך שנייה: ע"י ביצוע ההרצה מהטרמינל של סביבת הפיתוח pycharm ע"י הרצת הפקודה: py main.py

מסכים:



ישנו דף עם תצוגה גרפית, נוחה וידידותית למשתמש, שמריצה 10 פעמים ברצף SOM על תוצאות הבחירות בכנסת ה24 (לפי קובץ הקלט שקיבלנו), כך שבכל איטרציה מוצג במרכז המסך ה SOM כלומר הכוורת (ההקסגון) בגודל של 61 תאים (שכאמור בסוף התהליך כל תא מייצג מספר ישובים). כפי שניתן לראות הכוורת בנויה בצורה ברורה תוך מתן דגש על החלק העיצובי (מבחינה ויזואלית).
לבסוף הצגנו את הפתרון הטוב ביותר כלומר את הרשת ההקסגונית (הכוורת) הטובה ביותר.
כמו כן, הוספנו פיצ'ר נוח, ידידותי ומעוצב של סרגל מידע בחלקו הימני עליון של המסך המכיל מקרא בין הצבע של תא לבין המצב הסוציו אקונומי שאותו הצבע מייצג, ובכך המשתמש יוכל להבין בצורה מרבית את משמעות הפלט ומשמעות הרשת ההקסגונית.
בנוסף, הוספנו בחלקו השמאלי עליון של המסך מקבץ נתונים מועילים ורלוונטיים למשתמש כגון: מספר הפתרון (מבין 10 הפתרונות), המרחק של הפתרון הטוב ביותר שהיה עד כה, מרחק הפתרון הגרוע ביותר שהיה עד כה, והמרחק הממוצע של כלל הפתרונות שהיו עד כה.
כמו כן בחלקו העליון של המסך כתוב בברור מהו מספר האיטרציה (בצבע כחול). בסוף כל איטרציה דאגנו שהלוח יעצור ויציג במרכז המסך את ה SOM (בכך המשתמש יוכל לראות את הדרך ולא רק את התוצאה הסופית) וע"י לחיצה על המשך נוכל להמשיך אל האיטרציה הבאה.
כאשר הריצה מסתיימת נפתח חלון עם הגרף אשר מכיל את כל 10 ההרצות השונות של אלגוריתם ה SOM שהצגנו כאשר מוצג המרחק הממוצע (של ווקטורי הdata מהווקטור של התא אליהם הם מופו) פר איטרציה (עבור כל הרצה של SOM שהרצנו מתוך 10 ההרצות).

קצת על הפרויקט:

יצרנו מעיין אפליקציית desktop אשר מיישמת SOM על תוצאות הבחירות בכנסת ה24 וזאת לפי קובץ הקלט שקיבלנו, בו נמצאות תוצאות הבחירות עבור כ 200 ישובים (197 ישובים בדיוק).
את היישובים מפינו לרשת ה SOM הקסגונלית, אשר הינה בגודל של 61 תאים על פי דפוסי ההדפסה, כך שבסוף התהליך כל תא מייצג מספר ישובים.
למעשה רשת ה SOM הינה רשת נוירונים פשוטה שתפקידה לעזור בביצוע ובעיקר בהצגת הקבצות (clustering).
רשת זו מסתמכת על מידע לא מונחה, כלומר unsupervised learning, ובעזרתו יוצרת מיפוי של מימד גבוה של קלטים רציפים לממד פלט נמוך ובדיד. כיוון שרשת זו הינה רשת SOM הקסגונלית, אזי צורתה היא מעין כוורת כך שכל תא בכוורת מייצג node.
לכל node (תא) יצרנו את התכונות הבאות:

- וקטור שמאותחל במספרים אקראיים ומשויך לתא בכוורת. נשים לב כי אורך הווקטור הינו כאורך הווקטור המייצג את דוגמאות הקלט.
- רשימת שכנים של התא (כלומר של ה node). אסביר בפירוט כיצד נקבעת רשימת השכנים בהמשך.
- מיקום במערך של המערכים כלומר index שמייצג את התא הנ"ל.

ייצוג רשת ה SOM ההקסגונלית (ייצוג הכוורת - הקסגון) :
כדי לייצג את הכוורת, כלומר את רשת ה SOM ההקסגונלית, השתמשנו במערך של מערכים באופן הבא:
בתא הראשון (אינדקס 0) של המערך הגדול יצרנו מערך באורך 5 (כלומר מ j=0 ועד j=4 כולל).
בתא השני (אינדקס 1) של המערך הגדול יצרנו מערך באורך 6 (כלומר מ j=0 ועד j=5 כולל).
בתא השלישי (אינדקס 2) של המערך הגדול יצרנו מערך באורך 7 (כלומר מ j=0 ועד j=6 כולל).
בתא הרביעי (אינדקס 3) של המערך הגדול יצרנו מערך באורך 8 (כלומר מ j=0 ועד j=7 כולל).
בתא החמישי (אינדקס 4) של המערך הגדול יצרנו מערך באורך 9 (כלומר מ j=0 ועד j=8 כולל).
בתא השישי (אינדקס 5) של המערך הגדול יצרנו מערך באורך 8 (כלומר מ j=0 ועד j=7 כולל).
בתא השביעי (אינדקס 6) של המערך הגדול יצרנו מערך באורך 7 (כלומר מ j=0 ועד j=6 כולל).
בתא השמיני (אינדקס 7) של המערך הגדול יצרנו מערך באורך 6 (כלומר מ j=0 ועד j=5 כולל).

בס"ד

בתא התשיעי (אינדקס 8) של המערך הגדול יצרנו מערך באורך 5 (כלומר מ j=0 ועד j=4 כולל).

<u>קביעת השכנים של כל תא ברשת ה SOM ההקסגונלית:</u>

כיוון שייצגנו את ההקסגון (הכוורת) כמערך גדול שמכיל בתוכו מערכים (בגדלים משתנים כפי שהסברתי למעלה) אזי נסביר לגבי השכנים של כל תא בכוורת לפי חלוקה למקרים:

- 4 התאים הראשונים, כלומר תאים עם אינדקס 0 עד 3 (כולל), במערך הגדול: במקרה הנ"ל השכנים של כל תא יהיו בתאים: ימינה, שמאל , למעלה, למטה, למעלה-שמאלה, למטה-ימינה.
- התא החמישי, כלומר תא עם אינדקס 4, במערך הגדול: במקרה הנ"ל השכנים של כל תא יהיו בתאים: ימינה, שמאל , למעלה, למטה, למעלה-שמאלה, למטה-שמאלה.
- 4 התאים האחרונים, כלומר תאים עם אינדקס 5 עד 8 (כולל), במערך הגדול: במקרה הנ"ל השכנים של כל תא יהיו בתאים: ימינה, שמאל , למעלה, למטה, למעלה-ימין, למטה-שמאלה.

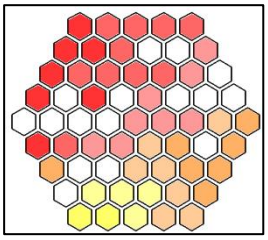
נדגיש כי עבור התאים שבדפנות של הכוורת (הקסגון) כמובן שהחוקים נשמרים כל עוד אכן קיים תא כזה, כלומר עבור תאים שבדפנות נצמדנו לחוקים הללו אך יחד עם זאת הקפדנו לא להביא תאים לא חוקיים (שאינם קיימים).

<u>שלבי העבודה:</u>

ראשית שמרנו את הקלט במטריצה כך שלקחנו רק את 13 העמודות השמאליות (המייצגות את כמות ההצבעות שהייתה פר מפלגה), ונרמלנו אותו ע"י חלוקה של כמות ההצבעות למפלגה מסוימת בכמות ההצבעות הכוללת שהייתה. (ביצענו את זה ע"י קריאה לפונקציה getMatrixData). כעת, יצרנו מערך של מערכים כך שבכל תא יש node (מכיל אינדקס, רשימת שכנים, ווקטור שאותחל תחילה עם ערכים רנדומליים בין 0 לערך המקסימלי שהיה באותה העמודה). כעת בתוך לולאה, שרצה כל עוד היה שינוי במיפוי של ווקטורים של ה data לתאים שבלוח ההקסגונאלי (כלומר כל עוד הפתרון לא הגיע למצב של התכנסות לפתרון הסופי), קראנו לפונקציה doFullEteration שמשייכת כל ווקטור שב data אל תא בכוורת (הקסגון), שהווקטור שמשויך לתא הנ"ל הכי דומה לו, כלומר בשלב זה עבור כל דוגמא שבקלט חיפשנו את הווקטור שדומה לה ביותר (שבאופן התחלתי היה כאמור אקראי) ואז "מקרבים" את הווקטור הזה לדוגמא ששויכה אליו. בנוסף גם את הווקטורים הנמצאים בסביבת הוקטור שנבחר (השכנים של ווקטור זה) מקרבים לאותה דוגמא, אם כי בשיעור מתון יותר סה"כ הקירוב מתבצע באופן הנ"ל: קירבנו ב 30% את הווקטור, עבורו הייתה דוגמא בקלט שהחליטה כי הוא הכי מתאים לה, אל הדוגמה ששויכה אליו, לאחר מכן קירבנו את השכנים של אותו ווקטור ב 20% (תאים במעגל הראשון מסביבו), ואז קירבנו את השכנים מרמה שנייה ב 10% (תאים במעגל השני). בסיום התהליך ימופו קלטים דומים לאזורים סמוכים במטריצה, והוקטורים המשויכים לאזורים אלו יהיו דומים במידה מסוימת לקלטים שאותם הם מייצגים.

כעת הלוח שלנו מעודכן ואנו מקווים שהוא תואם עד כמה שאפשר לסט של ה data (קלט). בשלב זה מפינו כל ווקטור של data לתא בכוורת ובכך אנו מציגים למסך בשלב הנ"ל את ההקסגון שנוצר לנו (נשים לב לשוני שכעת המיפוי מתבצע אחרי שהלוח עודכן והתקרב אל ה data). את רשת ה SOM הצגנו על ידי צבע שמתאר את הרמה הסוציואקונומית של הישובים שמופו לכל תא.

כלומר יצרנו 10 צבעים כך שאדום כהה מייצג מצב סוציו אקונומי 1, צהוב מייצג מצב סוציו אקונומי בינוני עם ערך של 5, ומצב סוציו אקונומי גבוהה מיוצג ע"י ירוק. כעת בכל דור עבור כל תא בהקסגון יצרנו ממוצע של כל הדרגות הסוציו אקונומיות של הערים (כלומר ווקטורים של data) שמופו לתא הנ"ל, הצגנו את רשת ההקסגון, ועבור כל תא לא ריק שבתוכו, צבענו אותו לפי הצבע שמתאים לממוצע הסוציו אקונומי אשר חישבנו עבור תא זה. נדגיש כי חזרנו על תהליך יצירת ה SOM 10 פעמים אשר בכל איטרציה מוצג למסך הSOM (הכוורת) ולבסוף הצגנו את הפתרון הטוב ביותר.



הפתרון הטוב ביותר הוא זה שעבורו סכום כל המרחקים של כל ווקטור מה data אל הווקטור השייך לתא אליו הוא מופה, הינו הקטן ביותר. **ניתן לראות כי בפתרון הטוב ביותר מתקבלת חלוקת צבעים של ממש בלוח אשר מתארת את המצב שציפינו לקבל: ערים בעלות מצב סוציו אקונומי זהה/דומה מצביעות לרוב לאותן המפלגות ביחס זהה של הצבעה למפלגות הנ"ל, ובכך כיוון שהווקטורים אשר מייצגים את ערים אלו דומים, אזי הווקטורים שלהם ממופים לתאים זהים/סמוכים בלוח ההקסגון ובכך בסופו של דבר אנו יכולים לראות זאת וויזואלית שאכן מתקבלת חלוקת צבעים בצורה ברורה.**

<u>סעיף א:</u>

חישבנו את המרחק בין ישוב בקלט לבין התא בגריד שמייצג אותו ע"י חישוב מרחק אוקלידי שכאמור עבור 2 הנקודות: (x_1, y_1, z_1) ו-(x_2, y_2, z_2) המרחק ביניהן מוגדר להיות : $\sqrt{(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2}$

<u>סעיף ב:</u>

אופן הקירוב הינו: הווקטור החדש = הווקטור (צנטרואיד) שהיה, אשר משויך לתא, כפול 0.7 ועוד הוקטור של הdata כפול ∝ (כך ש ∝ משתנה בהתאם לרמה), כלומר קירבנו את התא אליו מופה קלט מסוים לאותו הקלט ב 30% ולכן קיבלנו:

newVector = Oldvector*0.7 + dataVector*0.3

לאחר מכן קירבנו את השכנים של אותו ווקטור (תאים במעגל הראשון מסביבו) ב 20%

כך שקיבלנו newVector = Oldvector*0.7 + dataVector*0.2

ובאופן דומה, קירבנו את השכנים מרמה שנייה (תאים במעגל השני) ב 10% כך שקיבלנו:

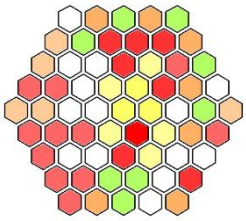
newVector = Oldvector*0.7 + dataVector*0.1

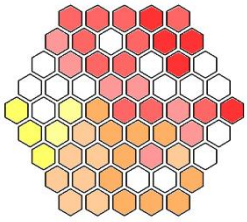
בס"ד

סעיף ג:

נשים לב כי ישנה חשיבות רבה לסדר בו אנו מציגים את הישובים למערכת. כשאנו קוראים לפונקציה doFullEteration, **הפונקציה רצה על ווקטורי הדוגמאות (data) ולמעשה משייכת כל ווקטור שב data אל תא בכוורת (הקסגון)** כך שהווקטור אשר משויך לתא הנ"ל הכי דומה לו. כלומר עבור כל דוגמא שבקלט חיפשנו את הווקטור אשר דומה לה ביותר **ואז קירבנו את הווקטור הנ"ל לדוגמא ששויכה אליו. בנוסף גם את הווקטורים הנמצאים בסביבת הווקטור שנבחר** (השכנים של ווקטור זה) **קירבנו לאותה דוגמא,** אם כי בשיעור מתון יותר. מכאן ניתן להבין בברור כי ישנה חשיבות לסדר בו אנו מציגים את היישובים למערכת וזאת כיוון שעבור כל דוגמא ביצענו עדכון לא רק לווקטור אשר הדוגמא שויכה אליו, אלה גם לכל שכניו (ממעגל ראשון ושני של הווקטור הנבחר) ולכן מובן כי כאשר נעבור לדוגמא הבאה בקלט מצב הלוח יהיה שונה לחלוטין ממה שהיה עבור הדוגמא הקודמת כלומר הכוורת תיראה שונה לחלוטין. דוגמא בולטת לכך אשר ממחישה את דברינו הינה בחירת הדוגמא הראשונה איתה נתחיל, שכן עבור הדוגמא שתבחר ראשונה, מצב הלוח (מצב הכוורת) הינו מאותחל עם ווקטורים אתחלתים (כיוון שהגדרנו את התאים להיות מאותחלים תחילה עם ווקטור שאותחל עם ערכים רנדומליים בין 0 ל ערך המקסימלי שהיה באותה העמודה) וכמובן שעבור הדוגמאות הבאות שיבחרו הלוח לא יכיל ווקטורים עם ערכים רנדומליים, אלה יכיל את הווקטורים לאחר שחלקם עברו "קירוב" אל הדוגמא הראשונה (כמובן לא בהכרח כל הווקטורים שבתאים השונים יעברו תהליך של קירוב, אך עדיין מדובר בכמות נכבדת של ווקטורים אשר כן יעברו קירוב לווקטור מה data). כלומר כאשר בחרנו דוגמאות בפונקציה doFullEteration (אשר בסופו של דבר תעבור על כל הדוגמאות), עבור כל דוגמא שנבחרה בכל פעם התבצע תהליך של קירוב לווקטור הנבחר אשר גורר עדכון של הווקטורים השכנים, שיתכן ובסיכוי גבוהה יתאימו בצורה מרבית לדוגמא הבאה שנעבור עליה ויעברו תהליך של קירוב עבור הדוגמא הבאה ובכך כל אחד גורר עדכון של השני ולכן זה משנה. לאור האמור לעייל, לא נרצה לעבור על כל הדוגמאות, בכל פעם, באותו הסדר שכן במידה וכן נעבור בכל פעם על כל הדוגמאות באותו הסדר נקבל מצב בו המערכת לומדת סדר ספציפי (כיוון שכל עדכון משפיע על השלב הבא) ולכן במצב הנ"ל המערכת תניב את אותן התוצאות בסופו של דבר ולא תלמד בצורה טובה. ניתן לראות הרצה כזו שביצענו בלוח שבצד ימין למטה. ניתן לראות שאין חלוקת צבעים ברורה כלומר המידע מעורבב. כיוון שישנה חשיבות רבה לסדר בו אנו מציגים את הישובים למערכת הבנו כי עלינו לבצע shuffle לכל הדוגמאות ורק אז קראנו לפונקציה doFullEteration אשר רצה על כל הדוגמאות, ממפה כל דוגמא בdata (כלומר כל דוגמא בקלט) לתא בלוח (הקסגון) ואז מקרבת אותו, ואת שכניו, אל אותה דוגמא ב data (הדוגמא שבקלט). כלומר, ניתן לראות כי כל עדכון גרר עדכון של השכנים ולכן אם נחזור על זה באותו הסדר ייווצר ביאס, לכן נרצה למנוע את המצב בכך שנבצע shuffle לפני כל קריאה לפונקציה שמעדכנת את הלוח ובכך מנענו את השגיאה הזו. ניתן לראות זאת בלוח שבשמאל למטה לאחר שכן ביצענו shuffle.

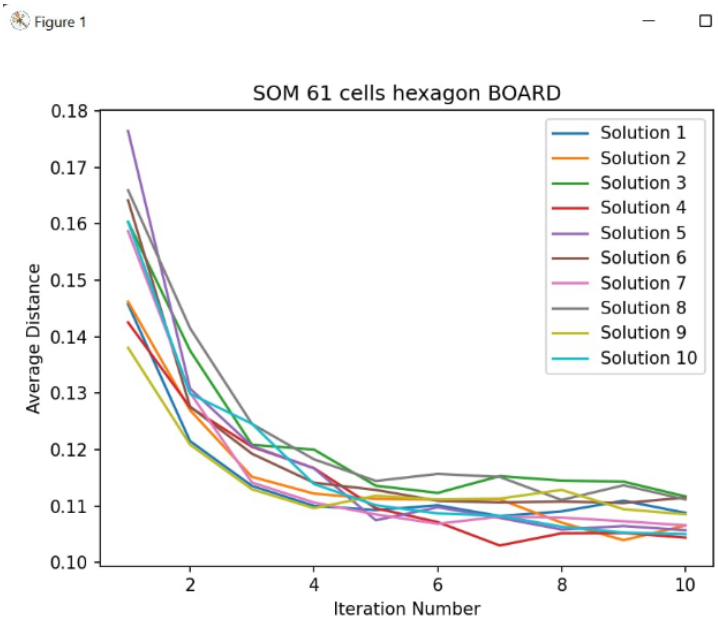
בלי shuffle <--





--> עם shuffle

סעיף ד:



כפי שהסברנו, חזרנו על תהליך יצירת ה SOM 10 פעמים כך שבכל איטרציה מוצג למסך הSOM (הכוורת) ולבסוף הצגנו את הפתרון הטוב ביותר. נשים לב כי לכל פתרון (עבור כל SOM שביצענו) חישבנו את הסכום של כל המרחקים, של כל ווקטור מהקלט (מה data) אל הווקטור בכוורת אשר שייך לתא אליו הוא מופה, לאחר מכן חילקנו את סכום זה בכמות הערים (דוגמאות) ובכך חישבנו את ממוצע המרחקים של הווקטורים שמייצגים את הdata מהווקטורים של התאים אליהם הם מופו. את ממוצע זה ייצרנו ושמרנו בכל איטרציה. לבסוף הדפסנו עבור כל אחד מ 10 הפתרונות את ערך המרחק הממוצע הנ"ל שהיה בכל אחת מהאיטרציות עד לסיום ריצת האלגוריתם, כך לדוגמא בגרף המצורף, ניתן לראות כי פתרון 4 קיבל את המרחק הממוצע הנמוך ביותר בסוף 10 האיטרציות – כלומר הפתרון שקיבלנו עבור הרצת אלגוריתם ה som בפעם הרביעית הינו הטוב ביותר אשר מתאר את הdata בצורה הצמודה והמדויקת ביותר מבין כל 10 הפתרונות, שכן הוא סיים במרחק ממוצע של כ 0.1. לאומת זאת, ניתן לראות כי פתרון מס' 3 הינו הפתרון הכי פחות מוצלח, כלומר הפתרון שמייצג בצורה הכי פחות טובה ומדויקת את ה data שכן קיבל ערך גבוהה יותר משל כל הייתר. כמו כן, אפשר לראות שיש פתרונות שפשוט התחילו בנק' פתיחה הרבה יותר טובה (שכן תחילה התחלנו את רשת ההקסגון עם ערכים רנדומליים), באופן זה, פתרון מס' 9 התחיל עם הערך הטוב ביותר של 0.14, ואילו פתרון מס' 5 התחיל עם הערך הכי פחות טוב של מרחק ממוצע של 0.18.

נזכיר כי את המרחק של ווקטור מהקלט אל הווקטור בכוורת (ששייך לתא אליו הוא מופה) חישבנו ע"י מרחק אוקלידי. ערך זה למעשה מהווה ומשמש כפונקציית הערכה למדידת טיב הפתרון, כאשר סכום מרחקים קטן יותר, פירושו פתרון טוב יותר אשר מייצג בצורה טובה יותר את ה data אותו רצינו ללמוד. לבסוף קבענו את הפתרון הטוב ביותר להיות הפתרון שעבורו הממוצע של כל המרחקים של כל ווקטור מהקלט (מה data) אל הווקטור בכוורת אשר שייך לתא אליו הוא מופה, הינו הקטן ביותר שכן זה אומר שהוא מייצג בצורה הטובה ביותר את הdata אותו רצינו ללמוד.