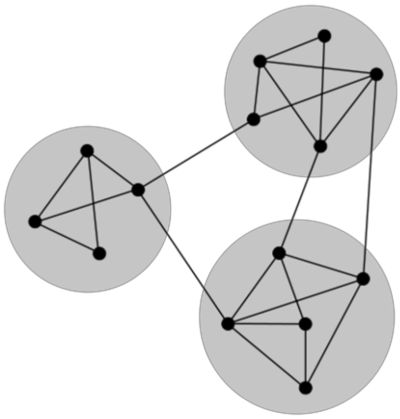
זיהוי חברי קואלציה ואופוזיציה בכנסת באמצעות שיטות של Spectral Graph Theory

Algorithms in Data mining - פרויקט סופי

[redacted]

הקדמה



אחד מהנושאים המעניינים ביותר בחקר רשתות מורכבות הוא Community Detection. כאשר בוחנים רשת, לפעמים ניתן לראות מבנים של קהילות – כלומר קבוצות של צמתים שהם יותר קשורים זה עם זה מאשר עם שאר הצמתים ברשת. המבנים הנ"ל יכולים לרמוז כי קיימת חלוקה של הרשת לקבוצות בעלות מכנה משותף.

לדוגמא: אם נסתכל על רשת המורכבת מצמתים המקבילים לחלבונים, ונגדיר קשת בין חלבון א' לחלבון ב' אם יש ביניהם אינטראקציה, אזי זיהוי של קהילות ברשת שכזאת, תוכל לרמז לנו על קבוצות של חלבונים שפועלים ביחד ע"מ לבצע פונקציה של התא.

Community Detection היא פעולה השונה מ-Clustering ; כשאנו רוצים למצוא Clusters, אנו בדר"כ מגדירים מראש את מספר הקבוצות שאנו רוצים לקבל – לדוגמא ב-K-means, אנו נותנים כ-Input את K – מס' הקבוצות הדרושות. לעומת זאת, אנו בדר"כ לא יודעים כמה Communities נמצאים ברשת, והאם בכלל יש כאלו. לכן, אנו מצפים כי אלגוריתם למציאת Communities, יהיה מסוגל בהנתן רשת, למצוא את ה-Communities ברשת, או להגיד כי אין כאלו.

ישנם מספר אלגוריתמים למציאת Communities בגרף. מאחר ובהרצאה האחרונה בקורס, התחלנו לדבר על Spectral Graph Theory, החלטתי כי הפרוייקט יעסוק באלגוריתם למציאת Communities המבוסס על Spectral Graph Methods.

האלגוריתם הנ"ל מופיע במאמר **Modularity and community structure in networks** , וכותב המאמר הוא **Mark Newman** אשר חוקר מבנים ופונקציות של רשתות.

כדי להדגים את יכולות האלגוריתם, החלטתי לבדוק האם הוא מסוגל לגלות מיהם חברי הקואליציה ומיהם חברי האופוזיציה בכנסת הנוכחית רק על פי גרף ובו הצמתים הם ח"כ וקשת בין ח"כ א' לח"כ ב' מסמנת עד כמה חברי הכנסת הנ"ל מסכימים על חוקים עליהם הם הצביעו ביחד. התוצאות שהתקבלו הן מרשימות ומעניינות ומופיעות במסמך זה.

האלגוריתם

המאמר ראשית מתמקד במקרה, בו אנו מעוניינים למצוא האם יש חלוקה טובה של הרשת לשתי קהילות בלבד. המטרה של האלגוריתם היא למצוא האם יש חלוקה טבעית של הצמתים ברשת לשתי קבוצות לא חופפות בגודל כלשהו.

אחת מהשיטות הכי טבעיות היא למצוא חלוקה של הצמתים לשתי קבוצות כך שהחתך מביא למינימום את מספר הקשתות בין שתי הקבוצות – min cut. השיטה הנ"ל לא מתאימה לבעיה שלנו, מאחר ואנו לא מעוניינים ב-partition של הגרף, אלא במציאת communities ולכן אנו לא יודעים מראש מהם הגדלים של הקהילות ואנו מרשים לקהילות להיות בכל גודל שהוא. לפיכך, גם הפתרון בו נשים את כל הצמתים בקבוצה א', וקבוצה ב' תהיה הקבוצה הריקה הוא אפשרי מבחינתנו, וברור כי הוא מביא למינימום את ה- min cut. אבל האם הוא יהיה הפתרון האינפורמטיבי ביותר לכל רשת? ברור שלא.

לפיכך, האבחנה במאמר היא שהשיטה של ספירת קשתות היא לא מתאימה בשביל לכמת את טיב החלוקה מבחינת ה-Community Structure. לעומת זאת, חלוקה טובה של רשת לקהילות היא כזאת, שמספר הקשתות בין הקהילות קטן מהמספר הצפוי של הקשתות בין הקהילות. כלומר אם מספר הקשתות בין שתי קבוצות הוא רק מה שהיינו מצפים לקבל באקראי, אזי לא ניתן לטעון כי יש כאן עדות לקיום של שתי קהילות. לעומת זאת, אם מספר הקשתות בין הקהילות הוא קטן בהרבה ממש שהיינו מצפים לקבל באקראי, או שמספר הקשתות בתוך קהילה הוא גבוה בהרבה ממה שהיינו מצפים לקבל באקראי, אזי יש הגיון בלהסיק שיש כאן חלוקה טבעית לקהילות.

הרעיון, שמבנה אמיתי של קהילות ברשת מאופיין בסידור של הקשתות שמפתיע מבחינה סטטיסטית, ניתן לכימות ע"י מדד שנקרא modularity. המדד הנ"ל, הוא עד כדי גורם כפלי, מספר הקשתות שנופלות בתוך כל קהילה פחות המספר הצפוי ברשת מקבילה בה הקשתות נבחרו באקראי. ה-modularity יכול להיות שלילי או חיובי כאשר ערך חיובי יותר מעיד יותר על סיכוי גבוה יותר לקיום מבנה של קהילה ברשת. לפיכך, שיטה אפשרית למציאת מבנה של קהילה ברשת, היא למצוא חלוקה של הצמתים ברשת, שתביא למקסימום את ה-modularity.

את מדד ה-modularity ניתן לרשום באופן הבא:

$$Q = \frac{1}{4m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\left(s_i s_j + 1\right)$$

כאשר:

- m הוא מספר הקשתות בגרף ולכן 2m הוא סכום הדרגות בגרף
- A_{ij} הוא אחד אם קיימת קשת בין צומת i ל-j ו-0 אחרת
- k_i הוא הדרגה של הצומת ה-i
- s_i הוא אחד אם הצומת ה-i משוייך לקהילה הראשונה, ו-1- אם הצומת ה-i משוייך לקהילה השנייה.

נסביר את הנוסחה: ניתן לראות כי $\left(s_i s_j + 1\right)$ שווה ל-0 אמ"מ הצומת ה-i וה-j בקהילות שונות. ולכן, כפי שצויין קודם, ה-modularity הוא אכן מדד המתייחס למספר הקשתות הנופלות בתוך כל קהילה. נסתכל על כל זוגות הצמתים שנמצאים בקהילה הראשונה. עבור כל הזוגות הנ"ל מתקיים כי $\left(s_i s_j + 1\right)$ שווה ל-2. ולכן, אם אנו מסתכלים רק הצמתים הנמצאים בקהילה הראשונה אזי התרומה שלהם ל-modularity היא:

$$Q = \frac{1}{2m}\left(\sum_{ij \in community-1} A_{ij} - \sum_{ij \in community-1} \frac{k_i k_j}{2m}\right)$$

ברור כי הגורם $\sum_{ij \in community-1} A_{ij}$ הוא מספר הקשתות הנמצאות בקהילה הראשונה. נותר להסביר מדוע הגורם $\sum_{ij \in community-1} \frac{k_i k_j}{2m}$ הוא מספר הקשתות הצפויות בקהילה הראשונה. נסתכל על שני צמתים i ו-j הנמצאים בקהילה הראשונה. נרצה לראות מהו המספר הצפוי של קשתות בין הצומת ה-i לצומת ה-j בקהילה הראשונה. נסתכל על קשת היוצאת מהצומת ה-i בעל הדרגה k_i. מהי ההסתברות, שהצד השני של הקשת מחובר לצומת j שהדרגה שלו היא k_j. מאחר ויש לנו סה"כ 2m דרגות בגרף, אזי ההסתברות היא $\frac{k_j}{2m}$. לפיכך אם אנו מתסכלים על קשת שמתחילה באחת "מהכניסות" של הצומת i ושואלים מהו המספר הצפוי של קשתות בינה לבין צומת j נקבל כי הוא $\frac{k_j}{2m}$.
אבל מאחר ולצומת i יש k_i כניסות, אזי המספר הצפוי של קשתות בין צומת i ל-j הוא $\frac{k_i k_j}{2m}$.

לפיכך מספר הקשתות הצפוי בתוך הקהילה הראשונה הוא $\sum_{ij \in community-1} \frac{k_i k_j}{2m}$.

כמובן שאותו ניתוח תקף לקהילה השנייה. חשוב לציין, שאנו מסתכלים על מספר הקשתות הצפוי בגרף בו הקשתות נוצרות באקראי אבל עדיין משמרים את הדרגות של הצמתים מהגרף המקורי. במקור אחר, מצאתי כי כותב המאמר מציין שבעקרון ניתן להתעלם מדרגות הצמתים וליצור קשתות בצורה אקראית לחלוטין, אבל מבחינה פרקטית נמצא כי התוצאות יהיו לא טובות.

כעת נתחיל לדון בשאלה כיצד למצוא וקטור s – כלומר חלוקה של הצמתים לשתי קבוצות כך שנמקסם את ה-modularity Q.

ראשית נשים לב כי $2m=\sum_i k_i=\sum_{ij}A_{ij}$ ולכן מתקיים: $\sum_{ij}\left(A_{ij}-\frac{k_ik_j}{2m}\right)=0$. ומכאן נקבל כי:

$$Q=\frac{1}{4m}\sum_{ij}\left(A_{ij}-\frac{k_ik_j}{2m}\right)\left(s_is_j+1\right)=\frac{1}{4m}\sum_{ij}\left(A_{ij}-\frac{k_ik_j}{2m}\right)s_is_j$$

כעת אם נסמן $B_{ij}=A_{ij}-\frac{k_ik_j}{2m}$ אזי נוכל לכתוב את Qבכתיב מטריציוני:

$$Q=\frac{1}{4m}s^TBs$$

למטריצה B נקרא modularity matrix. יהי $\{u_i\}$ ה-eigenvectors של המנורמלים של B. לפיכך ניתן לרשום את s לפי הבסיס $\{u_i\}$: $s=\sum_i a_iu_i$ כאשר $a_i=u_i^T\cdot s$. נסמן ב-β_i את הערך העצמי ה-i של B, ולשם נוחות נניח כי $\beta_1\geq\beta_2\geq\ldots\geq\beta_n$ ומכאן נקבל:

$$Q=\frac{1}{4m}\sum_i a_iu_i^TB\sum_i a_ju_j=\frac{1}{4m}\sum_{i=1}^{n}\left(a_i\right)^2\beta_i$$

המעבר האחרון נובע, מכך ש-$\{u_i\}$ ו-$\{\beta_i\}$ הם הו"ע והע"ע של B. בנוסף, אמרנו כי $\{u_i\}$ מנורמלים. מאחר והו"ע הם אורתוגונליים אזי: $u_i^Tu_j=0\ \ if\ \ i\neq j$ ו-$u_i^Tu_i=\left\|u_i\right\|^2=1$. כעת, מאחר ו-$\beta_1$הוא הערך העצמי הגדול ביותר, ברור כי כדי למקסם את Q, יש לבחור כי $a_1=1$ ו-$a_i=0\,\forall i\neq 1$. ומכאן נקבל כי $s=u_1$ הוא הפתרון המביא למקסימום את Q. כלומר אם נבחר את s להיות הו"ע הראשון (שמתאים לערך העצמי הגדול ביותר) אזי נביא למקסימום את Q. אבל s הוא לא כל וקטור. הוא חייב לקיים כי $s_i\in\{-1,1\}\ \forall i$. וברור כי u_1

לא מקיים תכונה זאת בהכרח. ננצל את העובדה כי למדנו שכדי למקסם את Q, יש לבחור s שמקביל ל-u_1, ומאחר ואנו לא יכולים בהכרח לקיים זאת, נרצה כי s יהיה כמה שיותר מקביל ל-u_1. במילים אחרות, נרצה למקסם את $a_1 = u_1^T \cdot s$. מאחר ו-s הוא וקטור בו כל ערך הוא 1 או 1-, ברור כי כדי למקסם את $a_1 = u_1^T \cdot s$ יש לבחור כי $s_i = 1$ אמ"מ $(u_1)_i > 0$ ו- $s_i = -1$ אמ"מ $u_i < 0$.

לסיכום האלגוריתם הוא כדלקמן:

- חשב את המטריצה B
- מצא את הוקטור העצמי המתאים לערך העצמי הגדול ביותר – u_1
- לכל צומת i=1..n, החלט כי i נמצא בקבוצה הראשונה אם u_i>0 ובקבוצה השנייה אחרת.

<u>הניסוי</u>

כפי שנכתב בהקדמה, כדי להדגים את יכולות האלגוריתם, החלטתי לבדוק האם הוא מסוגל לגלות מיהם חברי הקואליציה ומיהם חברי האופוזיציה בכנסת הנוכחית רק על פי גרף ובו הצמתים הם ח"כ וקשת בין ח"כ א' לח"כ ב' מסמנת עד כמה חברי הכנסת הנ"ל מסכימים על חוקים עליהם הם הצביעו ביחד.

ע"מ לעשות זאת, השתמשתי באתר oknesset.org; האתר מכיל נתונים עבור כל הצבעה שנעשתה בכנסת הנוכחית. בין הנתונים: אילו ח"כ הצביעו בעד ואילו ח"כ הצביעו נגד.

אספתי את כל הנתונים לגבי כל ההצבעות מתחילת כהונתה של הכנסת הנוכחית, ועד ל-02/04/2012. באמצעות הנתונים הנ"ל חישבתי לכל שני ח"כ את הערכים הבאים:

pos[i,j] – מספר ההצבעות בהם הח"כ ה-i והחבר הכנסת ה-j הצביעו אותו הדבר (שניהם הצביעו בעד או שניהם הצביעו נגד).

neg[i,j] – מספר ההצבעות בהם הח"כ ה-i והחבר הכנסת ה-j הצביעו שונה (אחד הצביע בעד והשני נגד).

כעת הגדרתי משקל w לכל קשת (i,j) המתאימה לח"כ ה-i ולח"כ ה-j באופן הבא:

- אם הח"כ ה-i והח"כ ה-j מעולם לא השתתפו באותה הצבעה אזי $w(i,j) = \frac{1}{2}$.
- אחרת: $w(i,j) = \frac{pos[i,j]}{pos[i,j] + neg[i,j]}$.

ניתן לראות כי w היא מעין ההסתברות לכך שהח"כ ה-i והח"כ ה-j מצביעים אותו הדבר. אם הם מעולם לא השתתפו באותה הצבעה, אז אין לנו מידע לכאן או לכאן ולכן אנו מחליטים כי "ההסתברות" היא 0.5. אם יש לנו מידע עבורם, ניתן לראות כי ככל שמספר החוקים בהם חברי הכנסת ה-i וה-j הצביעו אותו הדבר גדול מאשר מספר החוקים בהם חברי הכנסת ה-i וה-j הצביעו שונה, כך $w(i,j)\to 1$. וככל שמתקיים ההפך, $w(i,j)\to 0$.

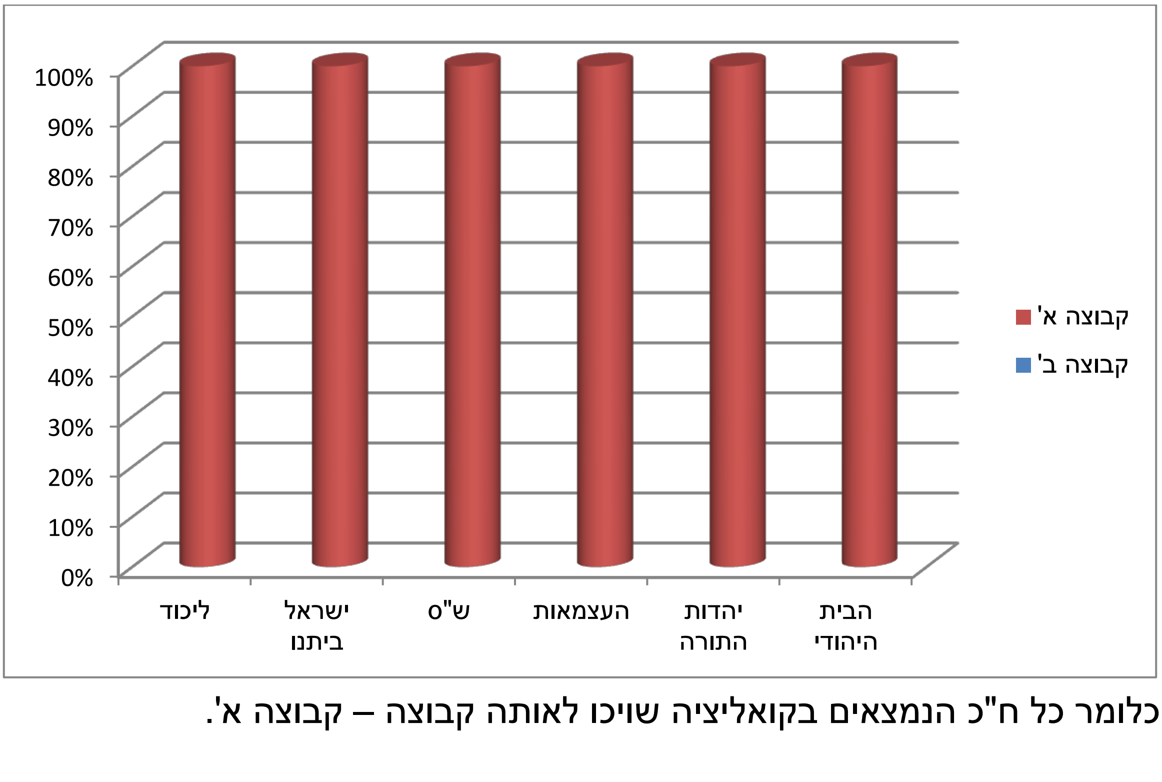
מאחר והאלגוריתם במאמר מתייחס ל-A_{ij} כמטריצה המכילה ערכי 1/0 (או שיש קשת או שאין) ואין התייחסות למשקל, החלטתי לשנות קצת את האלגוריתם כדי שיתאים לפי' המשקל שהגדרתי:

- $A(i,j)=w(i,j)$
- k_i – סכום המשקלים של כל הקשתות שקצה אחד שלהם הוא הצומת ה-i
- m – סה"כ משקלי הקשתות בגרף

חוץ מהשינויים הנ"ל, הרצתי את האלגוריתם המתואר במאמר as is על ה-dataset הנ"ל.

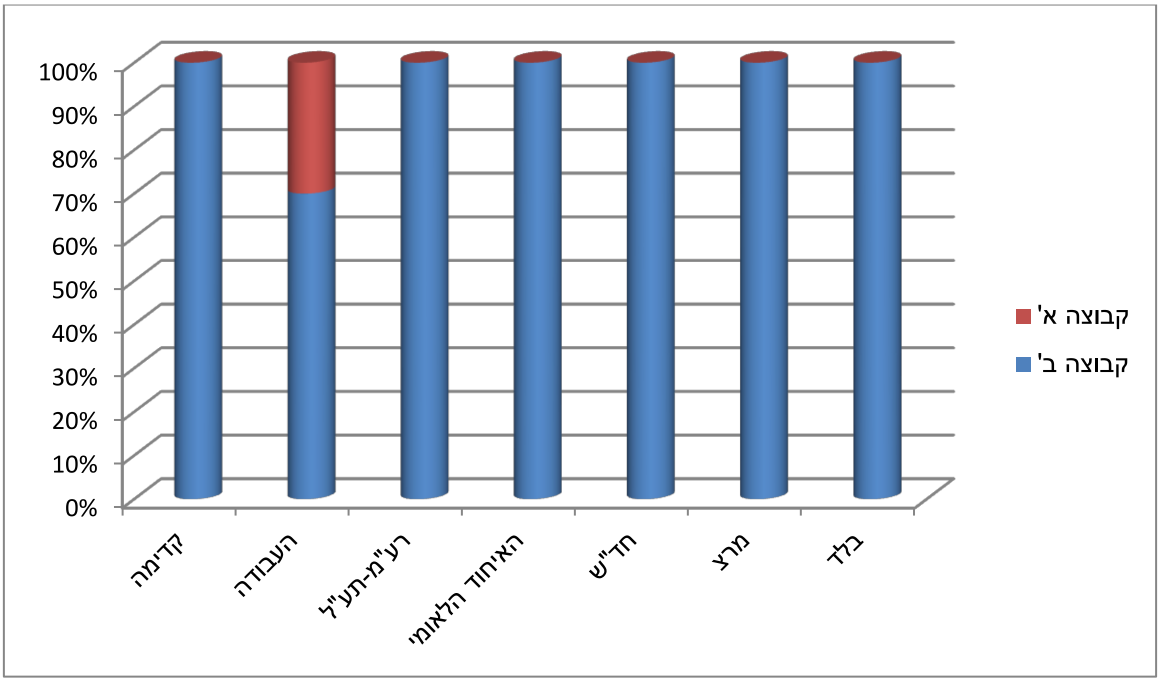
להלן התוצאות:

- עבור המפלגות הנמצאות בקואליציה:



כלומר כל ח"כ הנמצאים בקואליציה שויכו לאותה קבוצה – קבוצה א'.

- עבור המפלגות הנמצאות באופוזיציה:



כלומר כל ח"כ של כל המפלגות שנמצאות באופוזיציה למעט העבודה נמצאים בקבוצה ב'. כאשר 70% מח"כ של העבודה נמצאים גם הם בקבוצה ב' ו-30% מהם נמצאים בקבוצה א'.

לסיכום: קיבלנו כי קבוצה א' היא בעצם כל ח"כ בקואליציה ועוד 30% מחברי מפלגת העבודה. וקבוצה ב' היא בעצם כל ח"כ באופוזיציה פחות 30% מחברי מפלגת העבודה.

התוצאות הנ"ל מרשימות מאוד. ראשית, ניתן לראות כי האלגוריתם זיהה באופן ברור כי קיימות שתי קהילות שהן בקירוב רב האופוזיציה והקואליציה. ואם היינו מנסים לחשוב על חלוקה טבעית של ח"כ לשתי קבוצות אז קרוב לוודאי שהיינו בוחרים בחלוקה של קואליציה ואופוזיציה.

אבל בנוסף, נשים לב כי המפלגה היחידה שעבורה האלגוריתם כביכול "טעה" היא מפלגת העבודה. אבל האם הוא באמת טעה? נזכור כי הבחירות לכנסת הנוכחית נערכו ב- 10/02/09. לאחר הבחירות, מפלגת העבודה החליטה להצטרף בקואליציה ובעצם רק ב- 2011, כאשר אהוד ברק הקים את מפלגת העצמאות ונפרד ממפלגת העבודה, עברה המפלגת לאופוזיציה. לפיכך, העובדה כי האלגוריתם החליט כי חלק מחברי העבודה שייכים לקבוצה המזוהה עם הקואליציה היא רק טבעית לפי מהלך האירועים.

אם נבחן מיהם ח"כ ממפלגת העבודה שהאלגוריתם החליט כי הם שייכים לקבוצה המשויכת לקואליציה נראה כי אלו: יצחק הרצוג ,אבישי ברוורמן ובנימין בן אליעזר.

השלישיה הנ"ל היו שרים בממשלה עד ל-2011, ולכן הגיוני כי יהיו מזוהים עם הקואליציה. אציין כי כאשר ניסיתי להריץ את האלגוריתם שוב, הפעם רק על הצבעות שנעשו בחצי השנה האחרונה, כלומר לאחר שהעבודה פרשה מהקואליציה, קיבלתי כי שוב חלוקה ברורה של קואליציה ואופוזיציה כאשר הפעם כל ח"כ של העבודה שויכו לאופוזיציה. בנוסף, כל מפלגה למעט האיחוד הלאומי, שויכה נכונה לקואליציה ואופוזיציה. עבור האיחוד הלאומי שבמציאות שייכים לאופוזיציה, האלגוריתם החליט כי 50% מח"כ נמצאים באופוזיציה והשאר בקואליציה.

נשים לב לתכונה נוספת הנובעת מאופן פעולת האלגוריתם. כפי שהוסבר קודם, אנו מביאים את $a_1 = u_1^T \cdot s$ למקסימום ע"י כך שאנו בוחרים ערך $s_i=1$ היכן ש- $(u_1)_i > 0$ ו-$s_i=-1$ היכן ש-$(u_1)_i < 0$. ומאחר ולפי הניתוח שעשינו קודם, $a_1 = u_1^T \cdot s$ הוא הגורם שתורם הכי הרבה ל-Q, נובע מכך התכונה הבאה:

- אם נסתכל על כל הערכים $(u_1)_i$ כך ש- $(u_1)_i > 0$, נראה כי הערכים הגדולים ביותר מתאימים לח"כ שהיותם בקבוצה הראשונה הכי תורמת ל-modularity, ומאותה סיבה הערכים הקטנים ביותר מתאימים לח"כ שהיותם בקבוצה הראשונה הכי פחות תורמת ל-modularity.
- אם נסתכל על כל הערכים $(u_1)_i$ כך ש- $(u_1)_i < 0$, נראה כי הערכים הגדולים ביותר (בערך מוחלט) מתאימים לח"כ שהיותם בקבוצה השנייה הכי תורמת ל-modularity, ומאותה סיבה הערכים הקטנים ביותר (בערך מוחלט) מתאימים לח"כ שהיותם בקבוצה השנייה הכי פחות תורמת ל-modularity.

לפיכך ניתן להסתכל על כל קבוצה, ולראות מי הם ח"כ שהכי תורמים והכי פחות תורמים ל-modularity. ניתן לחשוב על ח"כ בעלי ערך $(u_1)_i$ גדול מאוד (בערך מוחלט) ככאלה שקשורים מאוד לקואליציה/אופוזיציה והעברתם לקבוצה השניה, היא מאוד לא כדאית ולא נכונה, ולעומת זאת ח"כ בעלי ערך $(u_1)_i$ קטן מאוד (בערך מוחלט) הם ח"כ שניתן להעביר אותם לקבוצה השנייה ללא השפעה גדולה על ה-modularity ולכן ניתן לראות אותם כח"כ "מתנדנדים" שפחות קשורים לקבוצה אליה הם שויכו.

לדוגמא, עבור ה-dataset הנ"ל, אם נסתכל מיהו ח"כ בערך הערך $(u_1)_i$ הגדול ביותר (בערך מוחלט) ששוייך לקבוצה שמזוהה עם האופוזיציה נקבל כי זאת ח"כ חנין זועבי מבל"ד. אני מאמין כי רוב האנשים יגידו כי היא ח"כ שהכי לא סביר שתהיה בקואליציה הנוכחית, ולכן העובדה כי ערך ה-$(u_1)_i$ כ"כ גדול (בערך מוחלט) עבורה אכן תומך בעובדה הנ"ל.

חלוקה ליותר משתי קהילות

בתיאור האלגוריתם, הראיתי כיצד המאמר מתאר את החלוקה של הרשת לשתי קהילות. אבל מה אם הרשת מורכבת מיותר משתי קהילות? הגישה שהמאמר בחר בה, היא לבצע חלוקה חוזרת של שתי הקהילות שהתקבלו מהחלוקה הראשונה וכך להמשיך. נתאר כיצד לעשות זאת ומתי כדאי להפסיק לחלק.

ראשית יש לשים לב, שזה לא יהיה נכון, אחרי החלוקה הראשונה לשתיים, למחוק את הקשתות בין שתי הקהילות שנמצאו ואז להריץ את האלגוריתם מחדש לכל תת גרף. זאת מאחר והדרגות שמופיעות בהגדרת המדד של ה-modularity יישתנו אם הקשתות הנ"ל יימחקו, ולפיכך כל מקסימיזציה נוספת שנבצע, תהיה על מדד לא נכון.

במקום זאת, מגדירים את ΔQ שהוא התרומה למודלריות מחלוקה נוספת של קבוצה g בגודל n_g לשתי קבוצות.

נקבל כי: $\Delta Q = \frac{1}{4m}\left[\sum_{i,j\in g} B_{ij}s_i s_j - \sum_{i,j\in g} B_{ij}\right]$.

הסבר: אם נסתכל על התרומה של הקבוצה g למודלריות Q היא $\frac{1}{4m}\sum_{i,j\in g} B_{ij}$ וזאת מאחר וכל האיברים שעליהם אנו סומכים נצאים באותה קבוצה g. אם נחליט לחלק את הקבוצה g לשתי תתי קבוצות, נקבל כי התרומה למודלריות של Q תורכב משתי תתי הקהילות שנקבל מהחלוקה. לפי הפיתוח שהראינו קודם, התרומה הנ"ל היא $\frac{1}{4m}\sum_{i,j\in g} B_{ij}s_i s_j$. לפיכך, ההפרש בין המודלריות Q לאחר חלוקה נוספת לבין המודלריות Q ללא חלוקה נוספת היא אכן הביטוי שרשמנו ל-ΔQ. המטרה שלנו היא למצוא וקטור s (עבור האיברים הנמצאים בקבוצה g) שיביא למקסימום את ΔQ. אם נקבל כי עבור הוקטור שמצאנו, $\Delta Q < 0$ אזי נבין כי החלוקה הנוספת לא תרמה ל-modularity Q ולכן נפסיק לבצע את החלוקה בענף הנ"ל. מכאן שע"י הביטוי ל-ΔQ, מצאנו גם תנאי לעצירת החלוקה.

כעת, כיצד נמצא s שמביא למקסימום את ΔQ? היינו רוצים לכתוב את ΔQ גם בכתיב מטריציוני כדי שנוכל להשתמש באותה השיטה שמצאנו עבור חלוקה של רשת לשתי קבוצות בלבד. לפיכך, מבצעים מספר טריקים ע"מ שנוכל לעבור לכתיבה מטריציונית:

$$\Delta Q=\frac{1}{4m}\left[\sum_{i,j\in g}B_{ij}s_is_j-\sum_{i,j\in g}B_{ij}\right]=\frac{1}{4m}\sum_{i,j\in g}\left[B_{ij}s_is_j-\delta_{ij}\sum_{k\in g}B_{ik}\right]=$$
$$\frac{1}{4m}\sum_{i,j\in g}\left[B_{ij}s_is_j-\delta_{ij}s_i^2\sum_{k\in g}B_{ik}\right]=\frac{1}{4m}\sum_{i,j\in g}\left[B_{ij}s_is_j-\delta_{ij}s_is_j\sum_{k\in g}B_{ik}\right]$$
$$\frac{1}{4m}\sum_{i,j\in g}\left[B_{ij}-\delta_{ij}\sum_{k\in g}B_{ik}\right]s_is_j$$

התבססנו כאן על העובדה כי $s_i^2=1$ מאחר ו- $s_i\in\{-1,1\}$. ואת הסכום $\sum_{i,j\in g}B_{ij}$ "פירקנו" באמצעות הדלתא של Kronecker כך שיחולק בין איברי האלכסון (i=j). כעת ניתן להגדיר את המטריצה $B^{(g)}{}_{ij}=\left[B_{ij}-\delta_{ij}\sum_{k\in g}B_{ik}\right]s_is_j$ ונקבל כי ניתן לרשום את ΔQ באופן הבא:

$$\Delta Q=\frac{1}{4m}s^TB^{(g)}s$$

לפיכך ניתן כעת למצוא s שממקסם את ΔQ באותו האופן בו מצאנו פתרון שממקסם את Q.

אציין בתור הערה, כי ניסיתי לחלק את שתי הקבוצות שקיבלתי קודם – אופוזיציה וקואליציה לתתי קבוצות באמצעות האלגוריתם הנ"ל. גם החלוקה של האופוזיציה וגם החלוקה של הקואליציה נתנה ערך ΔQ שלילי ולכן לא תרמה ל-modularity. כותב המאמר מציין שמנסיונו האישי נדיר כי נמצא חלוקה טובה של רשת לשתי קבוצות, שלא ניתנת לחלוקה נוספת. הוא מראה דוגמא למקרה שכזה במאמר. והנה, גם אנחנו קיבלנו תוצאה שכזאת.

ניתן להסביר את התוצאה הנ"ל, בכך שבמציאות של היום, ח"כ בקואליציה מצביעים ע"מ לשרת את הישרדות הקואליציה ולכן מצביעים במשמעת קואליציונית גבוהה וללא הבדלה סיעתית בולטת. עובדה שמבחינת האלגוריתם, הקבוצה שמייצגת את הקולאיציה לא ניתנת לחלוקה נוספת מבלי לפגוע ב-modularity. ניתן להסיק מכך כי מפלגות שהיו בעלות מצע ואג'נדה ייחודית, מבטלות את עצמם כאשר המטרה העיקרית היא לשרת את הישרדות הקואליציה.

מימוש

את הקוד כתבתי ב-python:

knesset-datamining.py – עובר על האתר oknesset.org ומייצר את הקבצים הבאים:

- votesDict.p – קובץ המכיל את כל תוצאות ההצבעה בכנסת הנוכחית עד לתאריך 02/04/2012.
- membersParties.p – קובץ המכיל את המפלגות אליה שייך כל ח"כ.

את שני הקבצים שקיבלתי מהרצת הקוד הנ"ל, צירפתי לפרוייקט. כך שבמקום להוציא את המידע מהאתר מחדש, יהיה ניתן להשתמש במידע שהופק על ידי.

communities.py – מקבל כקלט את votesDict.p ואת memberParties.p, ומוצא חלוקה לשתי communities.

הקבצים הנ"ל משתמשים בספריות הבאות, שניתנות להורדה בחינם:

- cPickle
- mechanize
- numpy
- bs4 (BeautifulSoup 4)
- re

ביבליוגרפיה:

1) **Modularity and community structure in networks by M. E. J. Newman**
   http://www.pnas.org/content/103/23/8577.full.pdf+html
2) **Networks An Introduction: M.E.J Newman**