12. Graph Neural Networks (GNN)

אלגוריתמים של בינה מלאכותית משיגים תוצאות מרשימות במגוון עצום של תחומים. במובן מסוים ניתן לומר שההצלחה שלהם נובעת מהיכולת שלהם לתפוס קשרים ומורכבים וללמוד פונקציות או התפלגויות שמכלילות את המבנים האלה גם למקרים או דוגמאות שהאלגוריתם הלומד לא ראה. עם זאת, ישנם מספר תחומים בהם ההצלחה בולטת בצורה ניכרת – Vision, Text and Speech. הסיבה העיקרית להצלחה בתחומים אלה נעוץ בעובדה שיש סדר ומבנה ברור לדאטה:

- Vision: תמונה מיוצגת על ידי מטריצה של פיקסלים ויש משמעות לקרבה של פיקסלים אחד לשני (פיקסלים קרובים כנראה קשורים יותר אחד לשני מאשר פיקסלים רחוקים), מה שהביא לפיתוח התחום של רשתות קונבולוציה (ולאחר מכן גם שימוש בViT – Vision Transformer).
- Text: קטע טקסט מורכב ממשפטים, שבתורם מורכבים ממילים, כאשר יש משמעות לסדר של המילים והמשפטים. המשמעות שיש לסדר הובילה לפיתוח רשתות מבוססות רכיבי זיכרון (LSTM, GRU) ולאחר מכן מנגנון ה-Attention שנהיה הכלי העיקר בעיבוד שפה. אלגוריתמים אלו מנצלים את הסדר המרכיב את הטקסט המלא, ומסיקים ממנו קשרים בין חלקי הדאטה השונים.
- Speech: קטע קול גם הוא מורכב מרצף של סיגנלים שיש ביניהם קשר סיבתי ומשמעות לסדר בהם הם מופיעים.

כאשר נתון סט תצפיות מדומיין מסוים, ניתן להיעזר במאפיינים שלו על מנת ללמוד את האופן בו הוא בנוי ומורכב. בצורה יותר פורמלית, אם נתון סט תצפיות $S = \{\boldsymbol{x}, \boldsymbol{y}\}$, כאשר $\boldsymbol{x} \in \mathbb{R}^{n \times d}$ הינו אוסף של אובייקטים (למשל נקודות במרחב, אוסף של תמונות וכדו'), ו-$\boldsymbol{y} \in \mathbb{R}^n$ הינו אוסף של labels, אנו מעוניינים ללמוד פונקציה f וסט פרמטרים θ המקשרים בין אובייקט לתיוג שלו, כלומר $f(x_i|\theta) = y$. כאמור, בזכות האופי של הדאטה, ניתן להיעזר בהנחות מסוימות המאפשרות ללמוד את f ו-θ בצורה די פשוטה, על ידי שיטות אופטימיזציה ביחס לפונקציית המטרה המוגדרת כך: $\theta = \arg\min \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(f(x_i|\theta), y_i)$. בדומיין של תמונות למשל, f הינה רשת המורכבת ממספר שכבות קונבולוציה, ו-θ מייצג את משקולות הרשת. בזכות העובדה שבתמונות יש קשר בין פיקסלים קרובים – רשת קונבולוציה נלמדת f מצליחה לייצג באופן טוב את הקשר בין הדוגמאות לתיוגים שלהם וכך גם להכליל את עצמה גם לדוגמאות שלא נראו בזמן האימון.

ישנם דומיינים נוספים שגם בהם יש מבניות לדאטה, אם כי יותר קשה ללמוד את ההתנהגות כיוון שהמבניות פחות ברורה ויותר קשה למדל את ההתפלגות שלה. דוגמה לכך היא התחום של Time Series, בו מנסים לנתח התנהגות של סדרות בזמן, כמו למשל מחיר מניה. האתגר בתחום זה נעוץ בכך שהדאטה אינו מסודר בצורה מוגדרת היטב, מה שמקשה על הלמידה של ההתנהגות הכללית שלו וממילא גם על התחזית של איך הוא יראה בעתיד. אתגר דומה קיים גם בדאטה טבלאי, בו הרבה פעמים הקשר בין התאים השונים בטבלה לא מספיק ברור או כלל לא קיים. ישנם תחומים בהם הבעיה מחריפה, כיוון שהאופן בו הדאטה בנוי אינו מכיל מבניות או חוקיות ברורה. בכדי לאפשר למידה יעילה וטובה גם לתחומים אלה, יש צורך לייצג את הדאטה בצורה מוגדרת ומסודרת היטב, ואז להשתמש בידע של מבנה הדאטה על מנת לחפש בו קשרים וכללים. למעשה, יש המון מקרים בהם ניתן לייצג את הדאטה באמצעות גרף, ואז מנסים ללמוד את ההתנהגות של הגרף על פי הקשרים שבו. בפרק זה נסביר בקצרה מה זה גרף ומהם קשרים שיש בגרף, ולאחר מכן נראה אלגוריתמים של למידה על גרפים, המאפשרים ללמוד גם סיטואציות בהן המבנה של הדאטה לא ברור או לא קיים.

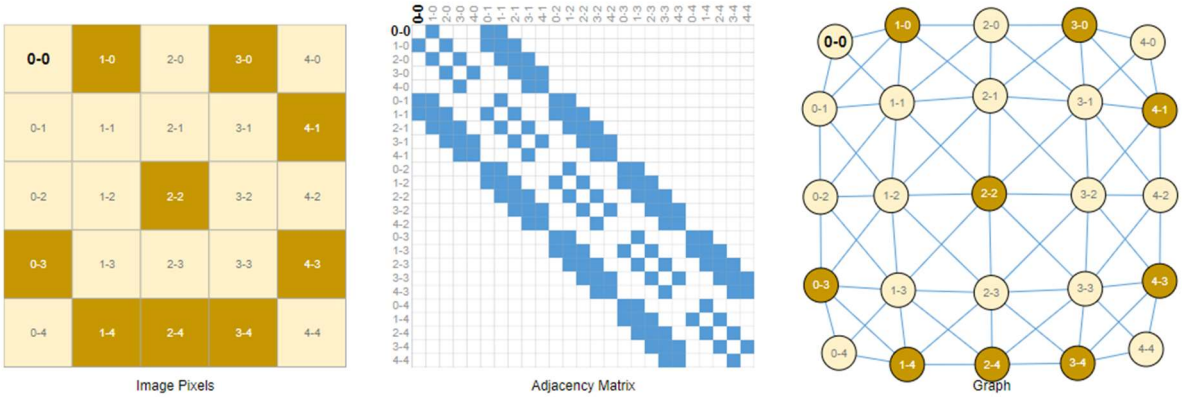
12.1 Introduction to Graphs

כאמור, גרפים יכולים לייצג מבנים מופשטים בתחומים רבים ומגוונים, וראשית נגדיר אותו בצורה פורמלית. גרף הוא ייצוג מופשט של קבוצה של אובייקטים, כאשר כל זוג אובייקטים בקבוצה עשויים להיות מקושרים זה לזה. האובייקטים הניתנים לקישור מכונים קודקודים או צמתים (Vertex), וקבוצת הקודקודים מסומנת באות V. הקישורים בין הקודקודים מכונים קשתות או צלעות (Edge) וקבוצת הצלעות מסומנת באות E. ניתן לייצג את הקודקודים באמצעות הצלעות באופן הבא: $E \subseteq V \times V$, כלומר כל צלע היא זוג הקודקודים, אותם היא מקשרת. גרף בעל קבוצת קודקודים V וקבוצת קשתות E מסומן באופן הבא: $G(V, E)$. ישנה הבחנה חשובה בין גרף מכוון, בו יש כיוון מוגדר לקשתות, לבין גרף לא מכוון בהם הקשתות הן חסרות כיוון. באופן פורמלי, בגרף מכוון קשת $e = (u, v) \in E$ הינה קשת היוצאת מקודקוד u ומגיעה לקודקוד v. בגרף בלתי מכוון אין משמעות לקודקוד ממנו הקשת יוצאת או אליו היא נכנסת, ולכן קבוצת הקשתות הינה $E \subseteq \{uv | u, v \in V\}$. ניתן לראות בגרפים בלתי מכוונים מקרה פרטי של גרפים מכוונים, בהם עבור כל זוג קודקודים u, v הקשתות מ-u ל-v ומ-v ל-u קיימות שתיהן, או חסרות שתיהן. ישנם עוד הרבה סוגים של גרפים והגדרות שונות, וכולן נשענות על ההגדרות הבסיסיות שהבאנו.

12.1.1 Represent Data as a Graph

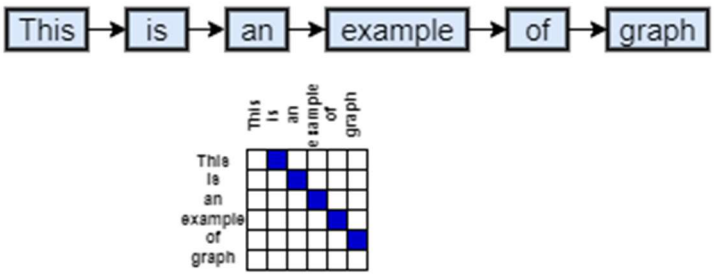
המבנה האבסטרקטי של גרפים מאפשר להם לייצג מגוון גדול מאוד של אובייקטים ומידע. דוגמה פשוטה ומוכרת לגרף היא רשת חברתית בה כל קודקוד מייצג משתמש ומכיל מידע לגביו, והקשתות מסמנות קשרים בין משתמשים שונים. שתי דוגמאות פחות טריוויאליות אך כאלה שיכולות להיות מיוצגות בצורה פשוטה על ידי גרף הן תמונות וטקסט, ובזכות המבניות שלהן אף ניתן להסיק מהגרף המייצג אותן כל מיני קשרים ומסקנות.

הדרך המקובלת לייצג תמונה היא באמצעות טנזור בעל שלושה ממדים – אורך (H), רוחב (W) ומספר ערוצי צבע (C). תמונה בעלת רזולוציה של 1024×1024 $(H = W = 1024)$ ושלושה ערוצי צבע $(C = 3)$ תהיה מיוצגת על ידי מטריצה $M \in \mathbb{R}^{1024\times1024\times3}$. ניתן לייצג תמונה גם באמצעות גרף לא מכוון באופן הבא: **הקודקודים** מייצגים את הפיקסלים כאשר כל קודקוד שומר את הערך של פיקסל יחיד, **והקשתות** מייצגות את הקשרים בין הפיקסלים, כאשר בין כל שני פיקסלים שסמוכים אחד לשני בתמונה המקורית תהיה קשת. אפשר להסתכל על גרף זה גם בעזרת מטריצת השכנויות שלו (adjacency matrix), שהינה מטריצה בגודל $\mathbb{R}^{|V|\times|V|}$, כאשר ערכו של התא A_{ij} הינו 1 אם יש קשת בין i ל-j (כלומר הם שני פיקסלים סמוכים), אחרת הוא 0.



איור 12.1

גם טקסט ניתן לייצג באמצעות גרף, אך הפעם מדובר בגרף מכוון, כיוון שכאשר מתייחסים למשפט, יש משמעות לכיוון בו קוראים את המילים. משפט בעל n מילים יכול להיות מיוצג באמצעות גרף בעל $|V| = n$ קודקודים ו-$|E| =$ $n - 1$ קשתות, כאשר יש קשת מכוונת בין כל מילה לבין המילה שבאה אחריה. כמובן שגם גרף זה יכול להיות מיוצג באמצעות מטריצת השכנויות, כאשר יהיה אלכסון (שמתחיל בתא $(1,0)$) של אחדות ושאר המטריצה תהיה אפסים.

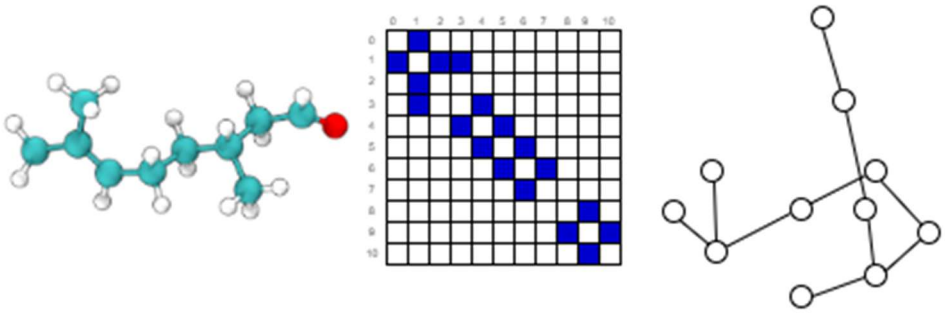


איור 12.2 ייצוג טקסט באמצעות גרף. כל מילה הינה קודקוד, ובין כל שתי מילים סמוכות יש קשת מכוונת.

עבור תמונות וטקסט הייצוג בעזרת גרף הוא לרוב לא הכי יעיל ואף מכיל יתירויות, ובדרך כלל משתמשים בייצוג הסטנדרטי שלהם (מטריצת פיקסלים או סדרת מילים), אם כי גם בתחומים האלה יש מצבים בהם גרפים יכולים להיות שימושיים, כפי שנראה בהמשך. לעומתם, יש כל מיני אובייקטים שאינם ניתנים לייצוג בצורה קומפקטית, ובדרך כלל הדרך היחידה לייצג אותם הינה באמצעות גרף. אובייקטים אלו בדרך כלל מאופיינים בכך שאין להם צורה מסודרת ומוגדרת מראש, מה שמתאפיין בגרף שיכול לקבל כל מיני צורות, כלומר, מספר משתנה של שכנים וקשתות לכל קודקוד (בניגוד לתמונה וטקסט, עבורם יש מבנה ידוע מראש של גרף). נתבונן על כמה דוגמאות:

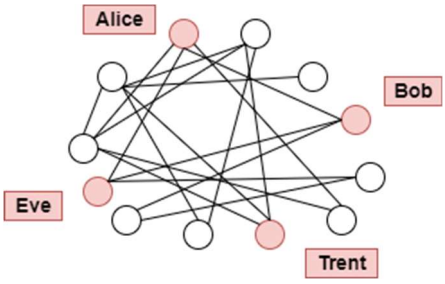
מבנה של מולקולות. מולקולות הן אבני הבניין של כל חומר, כאשר הן מורכבות מאטומים ואלקטרונים. כל החלקיקים מקיימים אינטראקציה משותפת, וכאשר יש זוג אטומים הנמצא במרחק יציב אחד אנו אומרים שהם מקיימים קשר קוולנטי. לזוגות שונים של אטומים וקשרים יש מרחקים שונים, כך שניתן לשרטט את המולקולה כאובייקט תלת ממדי.

ייצוג נח של אותו אובייקט הינו גרף בו כל קודקוד מייצג אטום וכל קשת מייצגת קשר קוונטי בין שני אטומים, כאשר אורכה של הקשת מסמל את המרחק בין שני האטומים.



איור 12.3 ייצוג מולקולת Citronellal באמצעות גרף – כל קודקוד מייצג מולקולה, וקשרים בין מולקולות מיוצגים באמצעות קשתות.

רשת חברתית. ברשת חברתית יש משתמשים, כאשר שני משתמש יכולים להיות מקושרים אחד לשני. כל משתמש יכול להיות מיוצג באמצעות קודקוד המכיל את המידע אודותיו, והקשתות מייצגות קשרים בין משתמשים – אם הם חברים באותה רשת אז תהיה ביניהם קשת בלתי מכוונת, ואם משתמש עוקב אחרי משתמש אחר, תהיה ביניהם קשת מכוונת.



איור 12.4 ייצוג רשת חברתית באמצעות גרף – כל קודקוד מייצג משתמש, וחיבורים בין משתמשים מיוצגים באמצעות קשתות.

ציטוטי מאמרים. כאשר רוצים לבנות רשימת מאמרים ולבחון כמה פעמים כל מאמר צוטט ומי ציטט אותו, ניתן לייצג כל מאמר על ידי קודקוד, וציטוט של מאמר על ידי מאמר אחר ייוצג על ידי קשת מכוונת (שיוצאת מהמאמר המצטט). באופן הזה מספר הקשתות הנכנסות לכל קודקוד שווה למספר הפעמים בהם המאמר צוטט על ידי אחרים.

יש עוד מגוון רחב של אובייקטים שנח לייצג באמצעות גרף (ולעיתים רבות זה מבנה הנתונים היחיד שיכול לייצג אותם), ולכל אובייקט יש מאפיינים שונים. לכל אובייקט יש מבנה אחר, כאשר חלק מהאובייקטים מיוצגים באמצעות גרפים צפופים (בהם מספר הקשתות גדול בהרבה ממספר הקודקודים) ויש אובייקטים המתאימים לגרפים דלילים (בהם מספר הקשתות הוא באותו סדר גודל של מספר הקודקודים). רשת חברתית למשל תהיה מיוצגת באמצעות גרף צפוף כיוון שלכל משתמש יש לפחות כמה עשרות או מאות חברים, ולכן מספר הקשתות גדול בהרבה ממספר הקודקודים. לעומת זאת, אם ממדלים מערכת כבישים בו כל צומת הינו קודקוד וכל כביש הוא קשת, הגרף יחסית דליל ומתקיים $|E| \leq c \cdot |V|$, כאשר c הוא לרוב בטווח שבין 2 ל-3.

12.1.2 Tasks on Graphs

לאחר שתיארנו בקצרה מהם גרפים ונתנו מספר דוגמאות של ייצוגי דאטה באמצעותם, נבחן את המשימות השונות שניתן לבצע על גרף. באופן כללי ניתן לומר שיש שלוש סוגי משימות ביחס לגרף: הסתכלות על הגרף בכללותו (Graph-level), הסתכלות על הקודקודים (Node-level), והסתכלות על הקשתות (Edge-level). כל משימה מתאימה לבעיות אחרות והגישות להתמודד איתן שונות זו מזו.

כשמתייחסים ל-Graph-level task, מעוניינים לקבל אמירה יחידה ביחס לאובייקט המיוצג על ידי הגרף. זו יכולה להיות שאלה בינארית (האם האובייקט מקיים תכונה מסוימת) או שאלה כללית יותר כמו למשל סיווג (לאיזו מחלקה שייך אותו אובייקט). ניקח למשל דוגמה של תמונה המיוצגת באמצעות גרף, אז Graph-level task יכול להיות שאלה לאיזו מחלקה האובייקט שייך.

לעיתים מתעניינים לא רק באובייקט באופן כללי אלא גם בפרטים שלו, ואז יש לבחון את הקודקודים והקשתות בפני עצמם. Node-level task היא משימה בה רוצים לאפיין כל קודקוד בפני עצמו. ברשת חברתית למשל, אם רוצים למצוא מאפיין מסוים אצל משתמשים (המיוצגים באמצעות קודקודים), אז יש לבחון כל קודקוד בפני עצמו ולבדוק

האם אותו מאפיין קיים אצלו או לא. דוגמה אחרת יכולה להיות משימת סגמנטציה של תמונה – אם תמונה מיוצגת באמצעות גרף וכל קודקוד הינו פיקסל, אזי במשימת סגמנטציה נרצה לשייך כל פיקסל למחלקה מסוימת.

ישנם מצבים בהם לא מספיק לבחון רק את הקודקודים עצמם ויש משמעות גדולה גם לקשרים ביניהם (הקשתות). משימה זו נקראת Edge-level task, וכאמור היא שמה דגש על הקשרים בין הקודקודים ללא תלות בקודקודים עצמם. נחזור שוב לדוגמה של תמונה, וכעת נתעניין בהבנת הסצנה המתרחשת בה. באמצעות סגמנטציה ניתן לשייך כל פיקסל (=קודקוד) לאובייקט מסוים או לקבוע שהוא חלק מהרקע, ובכדי להבין את הקשר בין האובייקטים השונים יש להסתכל על הקשתות שמחברות בין האובייקטים השונים. בצורה יותר פשוטה, ניתן להתייחס לכל אובייקט בתמונה כקודקוד ואז לבחון את הקשתות כמייצגות את היחס בין האובייקטים השונים. באופן הזה ניתן להבין את המתרחש בתמונה, באמצעות התמקדות בקשתות של הגרף.

12.1.3 The challenge of learning graphs

כאמור, ניתן לייצג באמצעות גרפים המון סוגים של אובייקטים, ויש מספר משימות שונות שניתן להתייחס אליהן. בכדי לבצע משימות על גרף, הדבר המתבקש הוא לבנות רשת נוירונים שמקבלת את הגרף ומנסה ללמוד את המאפיינים/פי'צרים שלו. אך מסתבר שזה לא דבר כל כך פשוט לביצוע, מכיוון שגרף בייצוג הבסיסי שלו הוא לא מבנה נתונים שאפשר להכניס באופן ישיר כ-input לרשת נוירונים. אמנם ניתן לייצר טנזורים של הקשתות והקודקודים ואותם להכניס לרשת, אך קלט כזה לוקה בחוסר של מידע בסיסי ומהותי לגבי המבנה עצמו של הגרף. במילים אחרות – הרשת יכולה לקבל טנזור של קודקודים יחד עם טנזור של קשתות, אך באופן הזה היא אינה יכולה לדעת כיצד הגרף בנוי בפועל, ומה המיקום היחסי של כל קודקוד ביחס לאחרים.

במקום להכניס לרשת רק את הקשתות והקודקודים, ניתן להכניס גם את מטריצת השכנויות, שמחזיקה מידע על היחסים בין מיקום הקודקודים השונים. גישה זו הייתה יכולה להיות טובה, אך יש בה שני חסרונות – אחד פרקטי ואחד מהותי. מבחינה פרקטית, ישנם גרפים עם המון קודקודים אך דלילים מבחינת קשתות, מה שיגרור מטריצה שכנויות מאוד גדולה שרובה אפסים, וזהו דבר מאוד לא יעיל מבחינה חישובית. חסרון נוסף ומהותי יותר נעוץ בעובדה שיש יותר ממטריצת שכנויות אחת לכל גרף, כאשר מטריצת השכנויות נגזרת מהסדר של הקודקודים. עבור גרף עם $|V|$ קודקודים, יש $|V|!$ אפשרויות לסדר אותם, וממילא מספר זהה של אפשרויות לבנות את מטריצת השכנויות. כיוון שהרשת אינה אינווריאנטיות לפרמוטציות של המטריצה, כלומר, שימוש במטריצות שונות כקלט לרשת יכול להביא לפלטים שונים, השימוש במטריצה זו הינו בעייתי ולא בהכרח יצליח לגרום לרשת ללמוד את הפיצ'רים של הגרף בצורה טובה.

בנוגע לבעיה הראשונה – ניתן להשתמש ברשימת שכנויות במקום במטריצה, מה שמייעל מאוד את החישובים והשימוש בזיכרון. כל איבר ברשימה הוא זוג של מספרים המייצגים את הקודקודים ביניהן יש קשת. באופן הזה חוסכים את כל האפסים שיש במטריצת השכנויות ומשתמשים ברשימה באורך מספר הקשתות. עם זאת, הבעיה השנייה עדיין עומדת בעינה, מה שמצריך גישה אחרת לייצוג גרפים כקלט לרשתות נוירונים.

ניתן להסתכל על הבעיה האחרונה גם מזווית אחרת. בפתיחה הזכרנו את העניין שיש תחומים בהם הדאטה מאורגן בצורה מסודרת וידועה מראש, כמו למשל תמונה המיוצגת כמטריצה של פיקסלים או טקסט שמיוצג כרצף של מילים הבאות אחת אחרי השנייה. אובייקטים אחרים, להם אין סדר ידוע, אמנם יכולים להיות מיוצגים על ידי גרפים אך עדיין יש בעיה בלייצג את אותם גרפים כקלט לרשת נוירונים באופן יעיל המאפשר ללמוד בצורה איכותית את הפיצ'רים של אותם אובייקטים.