מבוא לבינה מלאכותית – תרגול 9

נושאים:

הורדת ממדים לא לינארית:

- **(MDS מקובץ התרגול הקודם)**
- **Isomap**
- **Locally Linear Embedding (LLE)**
- **Laplacian Eigenmaps**

Isomap:

קיצור של Isometric Mapping. שלא כמו PCA ו- MDS, האלגוריתמים שעליהם נעבור היום מניחים שהדאטא המוטל לממד נמוך לא חי בתוך מרחב אוקלידי כי אם בתוך יריעה כלשהי (נניח, סתם דוגמה, על אליפסואיד). אם כן, על יריעה זו המרחקים מתנהגים אחרת מאשר במרחב אוקלידי, ונרצה שהמרחק שבין שתי נקודות יהיה המרחק הגאודזי שביניהן, בהתבסס על היריעה (אורך המסלול הקצר ביותר על היריעה שבין הנקודות). באלגוריתם זה נשתמש בגרף המבוסס על מטריצת המרחקים, ובקירוב של המרחק הגאודזי ע"י אורך המסלול הקצר ביותר בגרף.

<u>אלגוריתם:</u>

- בהתבסס על מטריצת המרחקים הידועה, בונים גרף כך שקודקודים יחוברו ל-k השכנים הקרובים ביותר אליהם ומשקלי הקשתות יתבסס על רכיבי המטריצה המתאימים.
- לכל זוג קודקודים בגרף, מחשבים כעת את הקירוב למרחק הגאודזי שביניהם. זה יהיה אורך המסלול הקצר ביותר בגרף בין קודקודים אלה.
- כאשר קיימים המרחקים הגאודזיים, אפשר להפעיל כל הטלה לממד נמוך מבוססת המרחקים החדשים, בפרט אלגוריתם classical MDS.

Locally Linear Embedding (LLE):

עוד דרך לבצע הטלה אל יריעה, הפעם יש לנו אוסף וקטורים במרחב מממד גדול מדי עבורנו. כמו Isomap, זה מתחיל מגרף מבוסס השכנים הקרובים ($k\ nearest\ neighbors - kNN$), אבל בזה בערך נגמר הדמיון.
השם נובע מתוך דרישה שההצגה של כל נקודה תהיה דומה לצירוף לינארי של ההצגות של שכניה של אותה נקודה.

<u>אלגוריתם:</u>

- כאמור, מתחילים מאוסף וקטורים מממד גבוה x_i, מהם בונים גרף מבוסס השכנים הקרובים ביותר (נסמן את שכני הדגימה ה-i בגרף ב-$N(i)$).
- נמצא מטריצת משקלים $W \in \mathbb{R}^{n\times k}$ לתיאור כל וקטור דגימה מקורי כסכום ממושקל של שכניה:

$$W = \arg\min_{\mathrm{W}} \sum_{i=1}^{n} \frac{1}{2} \left\| x_i - \sum_{j\in N(i)} W_{ij} x_j \right\|^2, \quad s.t. \sum_{j\in N(i)} W_{ij} = 1 \ \forall i$$

- בשימוש באותה מטריצה W, מחפשים את הייצוגים $y_1, \dots, y_n$ בממד הנמוך:

$$y_1, \dots, y_n = \arg\min_{y} \sum_{i=1}^{n} \frac{1}{2} \left\| y_i - \sum_{j\in N(i)} W_{ij} y_j \right\|^2$$

תחת אילוצים שדואגים שהפתרון לא יהיה טריוויאלי (y_i לא ייבחרו כולם אפסים), ומפורטים בשלמותם בהרצאה (בקצרה: דואגים שה-rank של מטריצת הפיצ'רים יישמר כגודל הממד אליו נטיל את הדאטא).

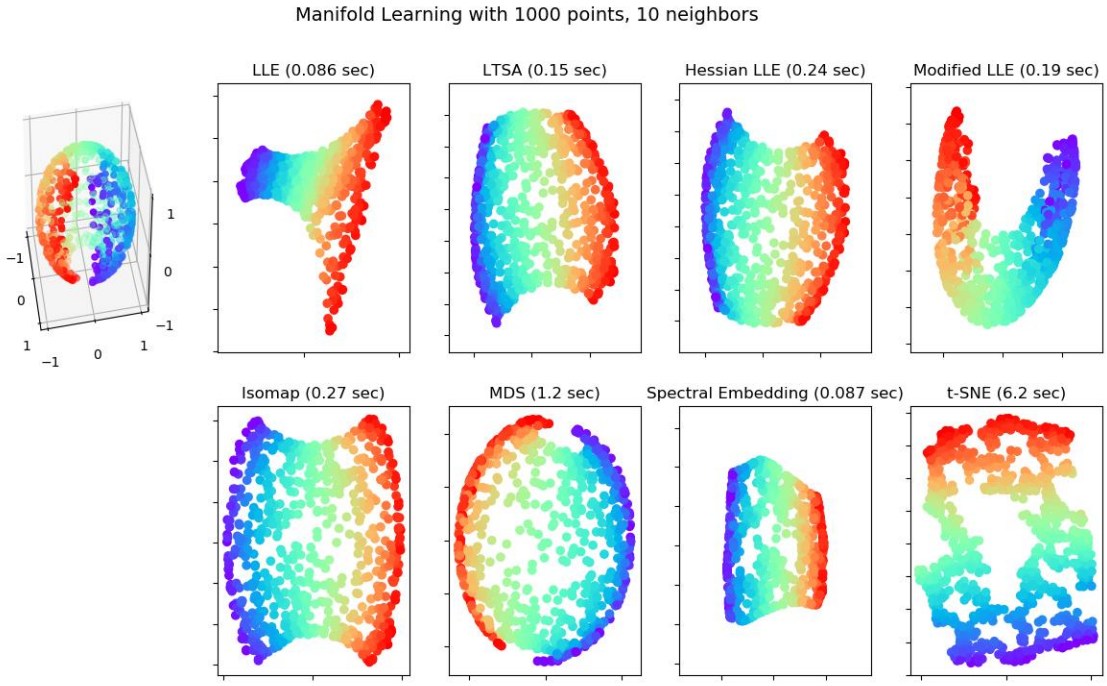
Laplacian Eigenmaps (ידוע גם בתור Spectral Embedding):

גם כאן עובדים עם גרף שנבנה מתוך מטריצת מרחקים של הדאטא לפני ההטלה (לא משנה איך בנינו אותו). הפעם משקל של קשת מייצג את רמת הדמיון שבין הקודקודים שהקשת מחברת. זה אלגוריתם שדומה מבחינת המימוש לאשכול ספקטרלי, ואכן בסוף מסתמכים על תכונות דומות של גרף לביצוע הטלה, כך שהווקטורים של קודקודים קרובים זה לזה יקבלו ערכים דומים.

<u>אלגוריתם:</u>

- בהינתן מטריצת משקלים שמייצגת גרף, $W \in \mathbb{R}^{n\times n}$, בונים את המטריצות הבאות:
  - מטריצה אלכסונית של דרגות ממושקלות D, כך ש- $D_{ii} = \sum_{j=1}^{n} W_{ij}$.
  - מטריצת הלפלסיאן של הגרף $L = D - W$ (אותה מטריצה מאלגוריתם האשכול הספקטרלי).
- מחפשים את מטריצת הווקטורים המוטלים $Y \in \mathbb{R}^{n\times d}$ כך ש- $2\ trace(Y^T LY)$ מינימלי, תחת האילוץ למניעת פתרונות טריוויאליים $Y^T DY = I$. זה מתבצע למעשה על ידי בעיית הערכים העצמיים הבאה:

$$L\vec{y} = \lambda D\vec{y}$$

כאשר הייצוגים שלנו יהיו למעשה ערכי d הווקטורים העצמיים המתאימים לערכים העצמיים הקטנים ביותר שאינם 0 (אז נקבל דברים טריוויאליים).

לסיכום האלגוריתמים בהם עסקנו היום (לא נראה בתרגול את אלגוריתם t-SNE, שגם הוא אלגוריתם להורדת ממדים, אבל בפועל משמש אנשים בעיקר לוויזואליזציה. השורה הראשונה מכילה וריאציות של LLE):