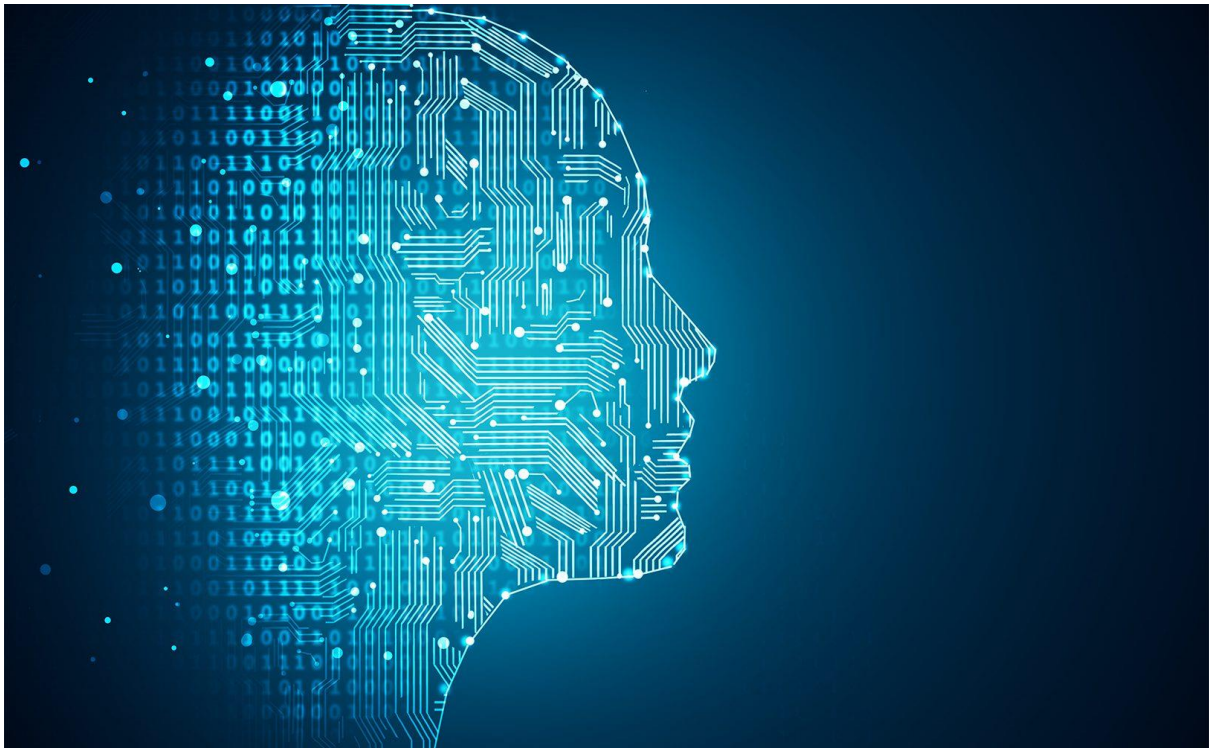
אוניברסיטת תל אביב

הפקולטה להנדסה

המחלקה להנדסת תעשייה וניהול

מבוא ללמידה עמוקה

דו"ח מטלת AutoEncoders – קבוצה 21



מוגש לידי:

ד"ר נעם קניגשטיין

מר יוני פוקס

מר יוסי לוי

מגישים – קבוצה 21:

מיה שני 206200149
גלעד יקואל 206035222
יובל זיו 207900283

מבנה ארכיטקטורת הרשת

1. **הגדרות המקודדים**

מבנה הencoder וכן הdecoder בנויים באותו אופן כאשר בחלק הראשון יש ירידה במימדים בסדר מסוים, והעליה במבנה לאחר הגעה למימד הנמוך ביותר, הינה באותו "סדר גודל" כפי שהורדנו את המימדים. בין ה encoder ל- decoder הוגדר dropout (כאקט רגולריזציה).

מימוש הרשת באופן זה הוחלט לאחר ביצוע מחקר בנושא, על בסיס התוכן אשר הוצג בהרצאות וכן חיפושים בפורומים מתאימים באינטרנט הנוגעים ביצירת רשתות AE[1].

כמו כן, בנוסף, בוצעו שינויים נוספים במימדים ובהסתברות לערך הdropout על מנת להגיע לביצועים הטובים ביותר במדד ה-Recall.

2. **פונקציית ההפסד (loss function)**

ה- loss function שנבחרה לשימוש הינה cosine similarity[2]. פונק' זו נבחרה לשימוש בשל מספר יתרונות:

ראשית, פונקציה זו מודדת "מרחק" בין שני וקטורים (ששווה למעשה לקוסינוס הזווית ביניהם).

כאמור, הרשת מיוצגת ע"י וקטורים ולכן פונקציית הcosine similarity מהווה פתרון מתאים:

הפונקציה מקבלת את וקטור A (הפרדיקציה) עם וקטור B (שהינו וקטור המטרה, התוצאה האמיתית של הרשומה); נעשה שימוש בייצוג פונקציית הcos, וככל שהוקטורים **דומים** יותר, כך הזווית ביניהם קטנה יותר – כלומר מדד הsimiliarity גבוה יותר (שואף ל1).

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

שנית, באמצעות שימוש בחבילת Pytorch ניתן לממש את המדד בקלות.

נשאף למקסם את מדד הsimiliarity או למזער את המדד של 1-similiarity (שכן פונק' מטרה יש **למזער**).

---

[1] מקור לחומר עזר: https://www.geeksforgeeks.org/implementing-an-autoencoder-in-pytorch/

[2] מקורות לחומר עזר: https://towardsdatascience.com/a-guide-to-neural-network-loss-functions-with-applications-in-keras-3a3baa9f71c5

https://towardsdatascience.com/understanding-cosine-similarity-and-its-application-fd42f585296a

3. **רגולריזציה**

נעשה שימוש ברגולריזציית bottle neck הקטן בסדרי גודל מהייצוג, יחד עם שכבת Dropout (בהסתברות p=0.5). הדבר נדרש כיוון שכפי שנלמד בכיתה, שימוש ברגולריזציה עשוי למנוע מצב של overfitting של המודל עבור סט האימון. כמו כן, הרגולריזציה יוצרת ספארסיות (במטריצות, ברשת הAE) ובכך משפרת את הליך האימון (יעילות חישובית משופרת).

בוצעו מספר ניסיונות לערכי p שונים (עבור הdropout) ולבסוף נבחר שימוש ב0.5 אשר הניב את התוצאות הטובות ביותר במדדי הבעיה.

4. **בחירת הארכיטקטורה**

בעת בחירת הארכיטקטורה לרשת, כפי שתואר בסעיף הראשון, היה ניסיון ליצירת רשת סימטרית במידת האפשר, תוך ניסוי וטעיה של שינוי מימדי השכבות השונות בתוך ה-Autoencoder. עד כדי התייצבות והגעה לתוצאה הטובה ביותר שהניב מדד ה-Sensitivity.

בדומה למה שהוצג בכיתה, נעשה שימוש בפונקציות ReLu בין שכבות הרשת על מנת לשמר את תכונת ה-Non-Linearity לצד יעילות סיבוכית המתבטאת בזמן ריצה נמוך.

<u>הליך האימון</u>

1. **יחס הפיצול בין סט אימון וולידציה**

אופן פיצול הדאטה לsets train / validation נבחר כך ש80% מהדאטה שויכו לסט האימון ו-20% הנותרים מהדאטה שויכו לסט הוולידציה (סדרי גודל פופולריים לחלוקה. למשל, שיטת K-Folds עם פרמטר K=5). תחילה נבחרה חלוקה של 10-90 לטובת סט האימון, אך בעת הרצת המודל הדבר הביא לתוצאות פחות טובות (overfitting בשלב מוקדם, ללא שיפור מהותי בהמשך האימון).

בוצעה הקטנה יחסית של סט האימון עד לעצירה על 80% מכלל סט הנתונים.

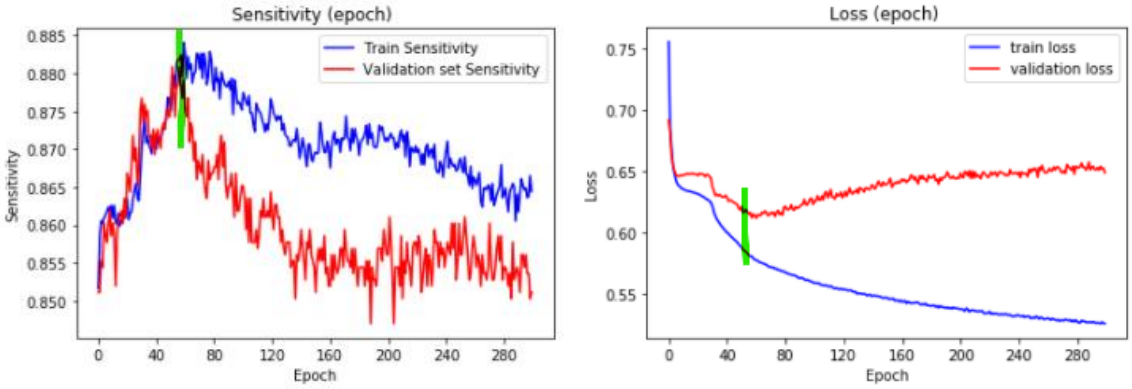
2. **"דגימות שליליות" – דגימת המוצרים השגויים**

בעבור כל משתמש נשמרו 2 מזהי מוצרים, כאשר כל אחד נלקח בצורה רנדומלית: האחד מתוך מוצרים אשר קנה (ערכים של 1) ואחד מתוך המוצרים שלא קנה (ערכים של 0).

בשלב הערכת המודל (validation) נבחן המודל על ייצוג המשתמש אשר <u>אינו מכיל</u> את המוצר שאהב, על מנת לזהות עד כמה המודל מכיר ולומד את טעם המשתמש, וכן מהי רמת הצדקה שלו בחיזוי המוצר המועדף על המשתמש (למעשה, שימוש במדד הRecall).

לאחר יצירת המטריקות, נקודת ה-Overfitting נקבעה לפי בחינה ויזואלית של עקומות ההפסד וה-Sensitivity כפי שניתן לראות בתרשים למטה (הנקודה מסומנת בקו ירוק).

לאחר מציאת מספר Epochs לאימון ללא חשש ל-overfitting (עצירה לאחר מספר ה epochs בהתאם לנקודה המודגשת בגרף), אומנה הרשת על כלל המשתמשים לצורך חיזוי סט ה-Test. כפי שניתן לראות בגרפים, משך האימון נבחר להיות 50 epochs.



3. **אפיון עדכוני ושגיאות ריצת המודל**

תחילה נבחר ביצוע חזרות של epoch 300 על מנת אימון הרשת ומציאת נק' התחלת הoverfitting (כפי שצוין בסעיף לעיל). לאחר צפייה ויזואלית בגרף, "נחתך" האימון לאחר epochs 50.

גודל ה-bach נקבע להיות 16. הסיבה לכך היא שמצד אחד כך הריצה לא תהיה ארוכה מדי (בתום כל batch נעשה עדכון) והרשת נמנעת מלהגיע למצב בו היא "לומדת" את כל המשתמשים בפעם אחת, ומאידך טובה מספיק על מנת לבצע למידה והכללה בזמני הרצה הגיוניים (מצב של למידת online- עדכון לאחר כל דגימה- היה "הורג" את זמן הריצה לחלוטין).

קצב הלמידה (ה-LR) נקבע להיות 0.0001, לפי הערכות מקובלות אשר הוצגו לנו במהלך ההרצאות, וכן לאחר ניסוי וטעיה תוך בחינת גרף ה-loss.

<u>מסקנות מאופן החיזוי</u>

אופן ביצוע החיזוי

לאחר בניית הרשת ואימונה, בוצע תהליך חיזוי באמצעות הרשת המאומנת. מומשה פונקציה שמטרתה לחזות מבין שני פריטים שהיא משווה ביניהם, מהו הפריט הסביר שהלקוח ירצה לקנות. בעת החיזוי, לכל מוצר מוענקת הסתברות לבחירתו על ידי המשתמש הנתון, ועל כן כתוצאה סופית מציגה הרשת את המקסימום מבין שתי ההסתברויות לכל מוצר.

בנוסף, בכל פעם נלקח batch של 2056 משתמשים (לצורך יעילות חישובית. אין טעם לשמור על גודל batch נמוך כיוון שלא נעשה עדכון בשלב זה, אלא תחזית בלבד), בוצע עבורם חיזוי ברשת ה AutoEncoders.

הערכת המודל

כפי שצוין לעיל, לאחר אימון של epochs 50 ושב מדד הrecall לכלל המשתמשים וערכו עמד על 0.891. מדד זה מעיד על כך שחיזוי המודל עובד בצורה יחסית טובה: כיוון שסט הנתונים דליל יחסית (ספארסי), כפי שנלמד בכיתה למודל הAutoEncoder ישנה העדפה לשחזר אפסים ובכך להגיע להגדיל את שגיאות התוצאה על גבי כלל סט הנתונים. מדד הrecall מעיד על כך שהרשת שנבנתה הצליחה להתמודד עם הבעיה בצורה טובה.

כמו כן, השאיפה לא הייתה לקבל מדד דמיון מושלם (loss = 0, similarity = 1) כיוון שהדבר אינו מעיד על ביצועי המודל עצמו.

בסה"כ, תוצאות המודל בעת המבחן על סט הנתונים היו טובות מאוד.

ההבדלים בין שני סט ה-test

1. **הצגת ההבדלים**

כאשר הורצו התחזיות עבור שני הסטים של ה-test, נבחנו מספר מטריקות שהוגדרו על מנת לקבל תמונת מצב (הערכה כלשהי) של המודל בעת ההרצות על הסטים (אימון המודל נעשה בצורה unsupervised, בשיטה של novelty detection- מאמנים את המודל להכיר דאטה נכון, ולא זיהוי דאטה שגוי תחילה):

a) מדד 1 - ממוצע גודל הפרש ההחלטה: בכל החלטה בחיזוי, נשמר ההפרש בין עוצמת הסיגנל המקסימלי למינימלי. בסופו של דבר, הוצג ממוצע ההפרש בין החיזויים, כלומר:

$$\mu = \frac{\sum|pred(Item1) - pred(Item2)|}{N}$$

כאשר N הינן סך התצפיות בכל סט מבחן נתון.

נראה כי עבור הRandom test התקבל מדד 0.173 לעומת סט הpopularity test שבו התקבל מדד 0.161. כלומר, בסט הפופולריות התקבל **הפרש נמוך יותר** – הגיוני כיוון שבסט זה הסיכוי שהמודל יתבלבל גבוה יותר, מאחר ומביאים דגימות פופולריות יחסית, והוא עשוי בטעות לשייכן למשתמש (על אף שמשתמש זה מעולם לא הזמין את המוצר). כלומר, מושקע מאמץ לבלבל את המודל ולגרום לו לשייך בחירות לא נכונות למשתמש הנבחן.

בסט הרנדומלי, אכן נצפה **לממוצע הפרש גבוה יותר** בין הסתברויות בחירת המוצרים ("העדפות המשתמש") כיוון שבסט זה בלבול המודל לא נעשה במכוון בעוצמה גבוהה כפי שנעשה בסט הפופולריות- הסיכוי שהמודל יטעה קטן, ועל כן ההפרש הממוצע בין הסתברויות הבחירה גדל.

המדדים שהתקבלו אכן מתכתבים עם לוגיקה זו, כפי שצוין לעיל.

b) מדד 2 - סטיית התקן של גודל ההפרש (פיזור): חישוב הפיזור של תצפיות המדגם בכל סט מבחן. ככל שהפיזור **נמוך יותר**, כך ההבדל בכל התצפיות בסט עבור המודל הופך לקטן יותר – הסיכוי של המודל להתבלבל בעת ביצוע החיזוי **עולה**. זאת כיוון שבעת הוספת מוצר פופולרי להשוואת המודל, סביר שההפרש בין שני המוצרים יהיה נמוך, והדבר ממשיך את קו ההסבר של מדד 1 (אשר ככל שישנה "תחרות צמודה" בין תחזית שיוך המוצר למשתמש בדגימה חדש, כך גם הפיזור מערך ממוצע שהינו נמוך ממילא קטן עוד יותר, סביב אותו ערך נמוך).

בהתאם לכך ולהסבר אשר נרשם במדד 1 (הרעיון זהה), יצופה שסטיית התקן בסט הpopularity test תהיה נמוכה יותר ("סט קשה יותר לפיצוח") וכן בסט הrandom test

סטיית התקן תהיה גבוהה יותר.
ואכן, התוצאות שהתקבלו במדד זה הינן 0.166 לRandom test וכן 0.158 ל Popularity test. התוצאות מתכתבות עם הלוגיקה שהוצגה לעיל.

c) מדד 3 - גודל ממוצע הסיגנל המקסימלי בעת ביצוע ההחלטה: המדד מחושב על גבי מיצוע ההסתברות הגבוהה יותר מבין השתיים בכל דגימה, ובכך מציג את "עוצמת ההחלטה" שנקבעה בממוצע למודל כאשר נדרש להחליט בין כל 2 אפשרויות לדגימות השונות בסט המבחן.
כיוון שכך, ערך הקרוב יותר ל1 יעיד כי בממוצע הרשת ביצעה את המלצתה באופן בטוח יחסית (אין הדבר מעיד על צדקת הרשת, אלא על עוצמת המלצתה בלבד).
מכאן נצפה כי בסט המבחן של הpopularity שבו קשה יותר לרשת להחליט (או לחלופין, סט המגדיל את סיכוייה לטעות בחיזוי) ערך הסיגנל יהיה גבוה יותר, כיוון שתמיד תוצג לפחות דגימה אחת אשר סיכוייה להיבחר אינם רעים. זאת לעומת הrandom test set, שבו עשויות להתקבל הסתברויות על בסיס מוצרים לא פופולרים, וכך הרשת תתן תחזית עם הסתברות נמוכה יחסית לבחירתם.
ואכן, התוצאות שהתקבלו במדד זה הינן 0.196 לRandom test וכן 0.219 ל Popularity test. התוצאות מתכתבות עם הלוגיקה שהוצגה לעיל.

2. **הצעת שינויים לשיפור ביצועים עבור הPopularity test**

לו הבעיה הייתה מוגדרת מראש לביצוע הבחנה על גבי סט הפופולריות, היו מוצעים השינויים הבאים:

a) הגדלת הbatch size- שינוי אפשרי הינו הגדלת גודל המקבץ לאימון (הגדלת הbatch size, אשר בעבודה זו נקבע להיות 16). הדבר עשוי לגרום לעדכון משקולות עם הסתברות החיזוי המתחשבת **בהעדפות של כלל המשתמשים** ובכך הרשת תלמד לזהות "טעם פופולרי" לעומת "טעם אישי", וכך הרשת תכיר את המוצרים הפופולריים טוב יותר ותלמד להבחין בין הטעם האישי של המשתמש לטעם הפופולרי.

b) שינוי הדאטה לאימון הרשת- **מדובר בביצוע סינון מקדים של כלל הדאטה לטובת אימון הרשת**.
בהינתן שקיים לצורך האימון מספיק רשומות, אפשרי להעניק ניקוד לכל מוצר לפי הפופולריות שלו על בסיס מספר הזמנות רב יותר למשל (כאשר יותר הזמנות = פופולרי יותר). כך נוכל לחשב **לכל משתמש** את "ניקוד הפופולריות" שצבר על ידי הזמנת המוצרים ולמיינם בסדר יורד. כעת ניתן יהיה להחליט על סף של סכום ציונים שהחל ממנו נחשיב את המשתמש בסט האימון, ומתחתיו המשתמש לא יכנס לסט האימון.
באופן זה רשת ה AutoEncoder תלמד לייצג את הטעם של המשתמשים שמייצגים הכי טוב את השימוש הפופולרי, וכך בעת חיזוי המוצרים הפופולריים ייתכן ויהיה לרשת כזו ייתרון על פני הרשת הקיימת.