הפקולטה להנדסה
ע"ש איבי ואלדר פליישמן
אוניברסיטת תל אביב

Information Retrieval and Recommender Systems

~HW2~

Id's:
312252794
313248544
207331752

הסברים:

פונק' get data:
לצורך אינדוקס מחודש "פייתוני" עבור ה-user_id וה-item_id ביצענו בדיקה על ה-ids המקוריים (ע"י הדפסה של nunique וmax על העמודות הרלוונטיות). מבדיקה על user_ids ראינו שהמספרים עולים באופן רציף מ1- עד n ועל כן החסרנו 1 מכל הערכים כך שה-ids ירוצו מ-0 עד n-1.
לפני שביצענו שינוי באינדוקס בעמודת users ב-validation, החלפנו כל item_id שלא קיים ב-train (כלומר שאין לנו כל מידע לגביו) ב-np.nan בכדי לא לאבד מידע על users. מבדיקה של item_ids ראינו שישנן "קפיצות" במספרים ולא קיימת רציפות ולכן עשינו שימוש בפונקציה replace. פונק' זו החליפה גם עבור ה-train וגם עבור ה-validation כל item_id ייחודי שנצפה בtrain באינדקס חדש כך שלבסוף כל הids נעים מ-0 עד n-1 (ה-replace בוצע כך ש-id ב-train מתואם עם id בvalidation). בנוסף בוצעה בדיקה האם ישנו user שקיים ב-validation אך לא ב-train כדי לדעת אם יש צורך לטפל בהתאם, אך התוצאה הראתה שאין מצבים כאלה.

simple mean-
בפונקציית fit איתחלנו מילון שמחזיק לכל user את הדירוג הממוצע עבורו. איתחול המילון נעשה ע"י שימוש בפונקציה groupby של pandas שפועלת על כל ה-data frame בבת אחת ולכן יעילה.
בחרנו להשתמש במבנה נתונים של מילון כיוון שהוא פשוט לשימוש, חסכוני בזיכרון ומהיר בשליפה.
בעת ביצוע חיזוי של user רק צריך לשלוף מהמילון את הערך המתאים לאותו user.

linear regression baseline-
נגזרות:

$$\alpha - learning\ rate, \qquad \gamma - gamma, \qquad \mu - the\ mean\ of\ all\ ratings$$

$$d'\ bu[user] = \alpha(2 * (rating(user, item) - \mu - bu[user] - bi[item]) + \gamma(2 * bu[user])$$
$$d'bi[item] = \alpha(2 * (rating(user, item) - \mu - bu[user] - bi[item]) + \gamma(2 * bi[item])$$

בכל epoch מעדכנים את הנגזרות לעיל לפי השורה בdata עליה רצים.
בסוף שמרנו את 3 הפרמטרים (ביאס של ה-user, ביאס של האייטם וביאס כללי) לקובץ pickle כנדרש.
עבור אייטם שקיים בvalidation ולא קיים בtrain, את החיזוי נבצע רק על פי הממוצע הכללי ועוד הביאס של היוזר כיוון שלא קיים לנו מידע על האייטם (מדובר ב-cold start) אך עדיין נרצה לייצר לו חיזוי כלשהו ולא לדלג על פרדיקציה זו.

knn-
מודל זה יחשב מדד דמיון בין כל זוגות הפריטים וישמור זאת במטריצה אשר תשמש אותנו לחיזוי. בחיזוי נבחר את K הפריטים עם מדד הדמיון הגדול ביותר עם הפריט עליו רוצים לבצע חיזוי ועליהם נעשה את החישוב המתאים.
עבור אייטם שקיים בvalidation ולא קיים בtrain, את החיזוי נבצע רק על פי ממוצע הדירוגים של היוזר כאשר הנימוקים הם כמו שהסברנו במודל הקודם. ראשית אנו מאתחלים מטריצה בה יש את המשתמשים בשורות, את הפריטים בעמודות ואת הרייטינג כערך ע"י שימוש בפונקציית pivot העושה את זה בבת אחת בצורה יעילה והופכים אותה להיות מטריצה ספרסית. עיבוד מקדים נוסף שעשינו הוא שמירת ממוצע הדירוגים שיש לכל פריט ולכל יוזר (ע"י group by בצורה יעילה על כל ה-data frame בבת אחת).
כאשר עוד לא בוצע fit ואין לנו מטריצת similarity אז עוברים על כל פריט שיש ולכל פריט נחשב מדד דמיון לפי פירסון תוך שימוש במתודה corrwith. בחרנו להשתמש במתודה זו כיוון שהיא משווה כל עמודה לעמודות הבאות שעדיין לא חישבה איתן דמיון ובכך מחשבת רק אלכסון עליון של המטריצה וחוסכת זמן ריצה כפול.
עבור זוגות אייטמים שמספר היוזרים שדירגו אותם נמוך מ-2 (אף יוזר או יוזר אחד) במקום קורלציה יחושב הערך לnan כיוון שאין משמעות לחישוב קורלציה במקרה הזה (בהמשך יהפוך לערך 0).
נרצה לשמור רק דמיון חיובי ועל כן כל דמיון שלילי נשנה את ערכו ל-0. בנוסף לכך דמיון של פריט עם עצמו הוא 1 ולא נרצה שייבחר בפריטים הכי דומים לצורך חיזוי ולכן גם את ערכו נשנה ל-0. מטריצת הדמיון האלכסונית לבסוף תהפוך

להיות ספרסית שכן הפיכת המטריצה לספרסית גורמת לייעול הקוד כך שאין שמירת דמיונות/דירוגים שהם 0, ובמערכות המלצה מטריצת משתמש-פריט מכילה המון תאים ריקים וכך גם מטריצת הדמיונות.
בנוסף נציין כי בשמירת הקובץ אין צורך לשמור דמיון ל-(X,Y) ו-(Y,X) שכן הם שקולים ולכן נשתמש במטריצה הספרסית האלכסונית שיצרנו, שכן כך נחסוך במקום האחסון.
כדי לבצע חיזוי נשמור מטריצה מלאה ולא אלכסונית אותה נייצר ע"י חיבור המטריצה האלכסונית עם ההופכית שלה.
לצורך החיזוי נשלוף את את הדמיון של הפריט עם שאר הפריטים ונוציא את K הפריטים עם הדמיון הגדול ביותר על ידי קבלת האינדקסים שלהם מהרשימה של הדמיון. לבסוף נחשב את החיזוי לפי הנוסחה כך שכדי לחשב את הערך החזוי בצורה יעילה וללא חזרה בלולאות ביצענו סכימה מצטברת של המונה והמכנה בנפרד ולבסוף חילקנו ביניהם והוספנו את הביאס. אם לא קיימים בכלל אייטמים דומים אז אין לנו דרך לחזות את הדירוג לפריט ועל כן נחזה בשלב הראשוני לפי ממוצע הדירוגים של הפריט.
בשמירת הערכים לקובץ עיגלנו 4 ספרות אחרי הנקודה כדי לחסוך במקום וכן שמרנו את האייטמים כ-Int16 ואת הדירוגים כ-float32.

<u>knn baseline-</u> מודל זה יחזה דירוג בצורה דומה ל-knn אך עם שינויים קלים בחישוב בהתאם לנוסחה. המודל ישתמש ב-biases שקיבלנו מה-linear regression baseline ומהממוצע הכולל שהוא שמר. בנוסף הוא ישתמש במטריצת הדמיון עבור זוגות הפריטים שחישבנו ב-knn. בצורה דומה גם כאן נבצע עיבוד מקדים כך שנשמור את ממוצע הדירוגים שיש לכל פריט ולכל יוזר ע"י ביצוע group by. בנוסף נשמור במטריצה ספרסית עקב יתרונותיה את כלל הדירוגים שיש עבור הפריטים והמשתמשים.
בביצוע ה-fit כדי ליצור טבלה חדשה של user to item השתמשנו בפונקציית pivot שעושה זאת בבת אחת מה שמייעל את השמירה שאינה צורכת מעבר על כל שורה.

<u>matrix factorization-</u>
נגזרות:

$$\alpha - learning\ rate, \qquad \gamma - gamma, \qquad \mu - the\ mean\ of\ all\ ratings$$
$$e\_ui = rating(user, item) - (\mu + bu[user] + bi[item] + (pu[user,:] \cdot qi[:, item]))$$
$$d'\ bu[user] = \alpha(e\_ui - \gamma * bu[user])$$
$$d'bi[item] = \alpha(e\_ui - \gamma * bi[item])$$
$$d'qi[:, item] = \alpha * (e\_ui * pu[user,:] - \gamma * qi[:, item])$$
$$d'qi[:, item] = \alpha * (e\_ui * qi[:, item] - \gamma * qi[user,:])$$

**הנגזרות חושבו לפי הנגזרות שהופיעו במאמר
את וקטורי הבייאסים איתחלנו לאפסים ואת מטריצות q וp איתחלנו למספרים רנדומליים כך שיהיה שוני בין מימד למימד (k מימדים). בכל epoch עידכנו את הנגזרות לעיל לפי השורה ב-data עליה רצים (העדכון על qi וpu בוצע בשורה אחת על כל k המימדים). בסוף הריצה על הepochs שמרנו תוצאה של מכפלת מטריצת qi במטריצת pu לצורך חישוב החיזוי אשר עושה שימוש במכפלה זו וב-bi,bu עבור user וitem ספציפיים.
גם במודל זה, עבור אייטם שקיים בvalidation ולא קיים בtrain, את החיזוי נבצע רק על פי הממוצע הכללי ועוד הביאס של היוזר כאשר הנימוק זהה למודלים הקודמים.

<u>ריכוז תוצאות ה-RMSE:</u>
נרכז את כל תוצאות ה-RMSE הסופי שקיבלנו עבור בחינת המודל על סט ה-validation עבור כל מודל.

שם המודל	simple mean	linear regression baseline	knn	knn baseline	matrix factorization
RMSE	1.0493	0.9582	1.0119	1.0119	0.9749