תרגיל בית 2 – מערכות המלצה

Simple mean

השימוש במערכת שמבוססת על ממוצע דירוגי היוזר. בשלב הפרדיקציה עבור (user i, item j) המערכת תחזיר את הממוצע של יוזר i ללא תלות בפריט j . מימוש האלגוריתם בצורה יעילה בוצע באמצעות שימוש ב group by של dataframe (מניח שמאחורי הקלעים ממומש בצורה מקבילית בסגנון map reduce). שמירת הנתונים לצורך שליפתם ב O(1) בוצעה באמצעות מילון כאשר המפתח הוא מספר היוזר והערך הוא ממוצע דירוגיו של היוזר.

רגרסיה לינארית

$$\text{Objective:}\min \sum_D (\widehat{r_{ui}} - r_{ui})^2 + \lambda \sum b_u{}^2 + \lambda \sum b_i{}^2$$

Update for b

$$b_u = b_u + \gamma(e_{ui} - \lambda \ b_u)$$
$$b_i = b_i + \gamma (e_{ui} - \lambda \ b_i) \ \ when \ \ e_{ui} = r_{ui} - \widehat{r_{ui}}$$

מודל שמחשב בנוסף לממוצע הכללי גם את הטיית המשתמש והטיית הפריט לפרדיקציה. על מנת למש זאת ביעילות נבנו 2 מערכים חד מימדיים . שימוש במערכים מאפשר שליפה ועדכון ב *O(1)* מכיוון שהאינדקסים במקרה זה הם מזהה היוזר ומזהה הפריט(מספר רץ).

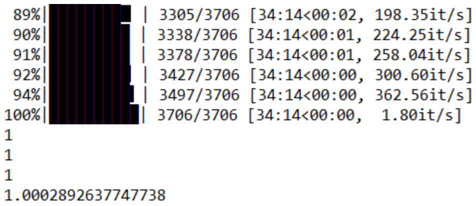
צעדי העדכון מתבצעים לאחר כל איטרציה(רשומה) שעבורה מחושבת שגיאת המודל ולאחר מכן צעדי העדכון שכל אחד מהם לוקח *O(1)* . התהליך הארוך הוא מעבר על כלל הדאטא כפול מספר האיטרציות הנדרש (*O(N*num_of_iters* . אך הוא בלתי נמנע לאור השימוש ב *sgd* .

KNN

מודל שמבוסס על חישוב סוג של מטריצה משולשת עליונה של הקורילציות בין הפריטים השונים ולאחר מכן שימוש בה כדי להעריך מה הציון שיתן יוזר לסרט על סמך סרטים בעלי קורילציה חיובית לסרט שהיוזר הזה כבר צפה בהם.

חישוב המטריצה דורש בניית מספר חישובי עזר. תחילה יש לבנות 2 מילונים אחד עבור היוזרים שהמפתח שלו זה מספר היוזר והערך הוא מילון עם ערכי *key value* שהמפתח שלהם הוא פריט והדירוג של היוזר עבור הפריט. אותו כנ"ל באופן סימטרי עבור הפריט. חישוב של מילונים אלו יאפשרו לנו לשמור את הדירוגים בצורה מצומצמת(ללא מטריצה גדולה שרובה המוחלט אפסים) עם גישה מהירה ב *O(1)* לערך שדירג משתמש פריט מסוים או לרשימת הפריטים אותם דירג או לרשימת היוזרים שדירגו פריט מסוים.

גם המטריצת קורילציות חושבה במבנה נתונים דומה(מילון שמכיל מילונים) רק הפעם המפתח של המילון הראשי היה פריט *i* והמפתח הפנימי היה פריט *j* עם ערך הקורילציה ביניהם. הדבר איפשר לשמור רק קורילציות חיוביות ורק כאלו שניתן היה לחשב אותם. כל קורילציה ניתן לשמור פעם אחת בקובץ לאור שיקולי סימטריה בחישוב הקורילציה. חישוב הקורילציות לקח בסה"כ 34 דקות בעזרת שימוש במבנה זה



```
 89%|█████████▊ | 3305/3706 [34:14<00:02, 198.35it/s]
 90%|█████████  | 3338/3706 [34:14<00:01, 224.25it/s]
 91%|█████████  | 3378/3706 [34:14<00:01, 258.04it/s]
 92%|█████████▏ | 3427/3706 [34:14<00:00, 300.60it/s]
 94%|█████████▍ | 3497/3706 [34:14<00:00, 362.56it/s]
100%|██████████| 3706/3706 [34:14<00:00,  1.80it/s]
1
1
1
1.0002892637747738
```

כמבוקש בתרגיל. חישוב הפרידיקציה הוא מורכב יותר מסעיפים קודמים ותלוי בכמות הפריטים שהיוזר המבוקש דירג וגם בעלי קורילציה חיובית לפריט שעליו נדרשת הפרידיקציה.

KNN BASELINE

זהה כמעט לחלוטין לקודם – טעינת הפרמטרים מהמטריצה ומהפיקל. יצירת הערכים הסימטרים במילון ולאחר מכן חישוב הפרידיקציה באותה רמת מורכבות כמו *KNN* .

MF

טיוב הפרמטרים נעשה ע"י חיפוש רנדומלי של ערכים אשר ע"י צירוף מיטבי שלהם יביא לתוצאה הטובה ביותר

אלו הפרמטרים שעבורם עשינו **hyperparameter** **tuning**: K epochs, gamma, lr

ולאחר חיפוש כזה אלו הפרמטרים שנבחרו:

matrix factorizationצעדי עדכון משתני

$$\widehat{r_{ui}} = avg + b_u + b_i + p_u {q_i}^T$$

$$e_{ui} = r_{ui} - \widehat{r_{ui}}$$
$$p_u = p_u + lr(e_{ui} \cdot q_i - \lambda \cdot p_u)$$
$$q_i = q_i + lr(e_{ui} \cdot p_u - \lambda \cdot q_i)$$
$$b_u = b_u + lr(e_{ui} - \lambda b_u)$$
$$b_i = b_i + lr(e_{ui} - \lambda b_i)$$

```
        lr=0.009030143369021236,
        gamma=0.0528855224903822365,
        k=44,
        epochs=46)
```

בדקנו את ההתכנסות בצורה הבאה (לא מתועד בקוד)

ממשיך לקטון אנחנו נכנסים למצב של אוברפיטינג train_rmse גדל שלוש פעמים ברציפות כאשר הvalidation_rmse ה
הגענו להתכנסות→

מבנה הנתונים שהשתמשנו בו הוא כמובן מטריצות ווקטורים לאור כמות הפעולות האלגבריות שנדרשות כאן. שימוש במעבד *GPU* יכול להריץ בצורה מהירה מאוד פעולות אופטימיזציה כאלו על וקטורים

rmse validation	אלגוריתם
1.04	Simple mean
0.9678	regression
1.000288633	knn
0.93	knn_baseline
0.889	MF