מסחר אלקטרוני – תרגיל בית 2 חלק 2

רקע

ספוטיקסט היא פלטפורמה חדשה לפודקאסטים, שמציעה תוכן מקורי הנגיש רק למנויים. בתחילת דרכה, לספוטיקסט אין פודקאסטים מוקלטים וגם אין מידע מוקדם על העדפות המשתמשים. עם זאת, בשלב ההשקה, שאורכו T שבועות, יש לה קבוצת משתמשים פעילים, והיא מעוניינת לרכוש את אמונם באמצעות המלצות מדויקות לתכנים מותאמים אישית. במהלך T שבועות, ספוטיקסט צריכה לקבל שתי החלטות בכל שבוע:

1. אילו פודקאסטים להפיק (בכפוף למגבלות תקציב ותשלום ליוצרים).
2. אילו פודקאסטים להמליץ לכל משתמש.

ההנאה של המשתמשים מהתוכן היא בינארית (נהנה / לא נהנה), ואינה ידועה מראש. מטרת הפלטפורמה היא למקסם את סך ההנאה של המשתמשים מההמלצות שקיבלו לאורך כל התקופה.

תיאור פורמלי של הסביבה והאינטראקציה

לספוטיקסט N מנויים $(1, \dots, N)$, ואפשרות להפיק K פודקאסטים שונים $(1, \dots, K)$. נניח שללקוח n יש הסתברות $P_{n,a}$ להנות מפרק של פודקאסט a, והיא קבועה בכל נקודת זמן. הסתברות זו אינה ידועה בתחילת האינטראקציה.

בכל שבוע, ספוטיקסט צריכה להחליט לאילו פודקאטים היא רוצה להפיק פרק נוסף, תחת אילוצי תקציב. עבור הפודקאסט ה-i, עלות הקלטת פרק חדש היא C_i, והיא לא משתנה. נסמן ב-$K^{(t)}$ את הפודקאסטים שספוטיקסט בחרה להוציא בשבוע ה-t. בכל שבוע, לחברה אסור לחרוג מתקציב של B, ולכן $\sum_{a \in K^{(t)}} C_a \leq B$. תקציב שלא השתמשו בו "בוזבז" ואינו נלקח בחשבון בפונקציית המטרה הסופית.

כעת, הפלטפורמה צריכה להמליץ לכל מנוי על פודקאסט אחד מתוך $K^{(t)}$. נסמן ב $a_n^{(t)} \in K^{(t)}$ את הפודקאסט שספוטיקסט ממליצה ללקוח ה-n שבוע ה-t.

נסמן ב-$X_n^{(t)}$ אינדיקטור שאומר אם הלקוח ה-n אהב את הפודקאסט שהמליצו לו בשבוע ה-t:

$$X_n^{(t)} = \begin{cases} 1 & w.p. \quad P_{n,a_n^{(t)}} \\ 0 & w.p. \quad 1 - P_{n,a_n^{(t)}} \end{cases}$$

בסוף השבוע ה-t, ספוטיקסט צופה ב $X_n^{(t)}$ לכל $n = 1, \dots, N$, וצריכה להחליט אילו פודקאסטים להפיק בשבוע ה-$t+1$ (עד $t = T$).

כפי שהוסבר, מטרת ספוטיקסט היא למקסם את כמות המנויים שאהבו את ההמלצות:

$$R = \sum_{t=1}^{T} \sum_{n=1}^{N} X_n^{(t)}$$

לסיכום - בכל שבוע t עליכם לבחור סט פודקאסטים להפיק באותו שבוע ($K^{(t)}$) תחת אילוץ התקציב, ולהמליץ לכל משתמש $n = 1, \dots, N$ על פודקאסט מבין הסט שבחרתם ($a_n^{(t)} \in K^{(t)}$), על סמך הפידבק מהשבועות הקודמים $X_n^{(t')}$, $t' = 1, \dots, t-1$, $n = 1, \dots, N$.

תיאור המשימה

מטרתם לממש מערכת המלצה שתתמוך בדרישות המתוארת – בחירת פודקאסטים להפקה ומתן המלצות ללקוחות.

מכיוון שהמטריצה P אינה ידועה, הקלט של המערכת לפני תחילת השבוע הראשון הוא:

- מספר השבועות T
- מספר המשתמשים N
- מערך של מחירי ההפקות $C_1, \dots, C_K$
- מגבלת התקציב B

הפלט של מערכת ההמלצה בתחילת השבוע ה-t:

- מערך של המלצות $a_1^{(t)}, \dots, a_N^{(t)}$ המהווה סט פיזיבילי תחת מגבלת התקציב:

$$\sum_{i \in \{a_1^{(t)}, \dots, a_N^{(t)}\}} C_i \leq B$$

הקלט של מערכת ההמלצה בסוף השבוע ה-t:

- מערך של תגובות הלקוחות $X_1^{(t)}, \dots, X_N^{(t)}$ כאשר כל $X_i^{(t)}$ הוא בינארי

ביצוע המערכת שלכם יממדו לפי תוחלת סך ההנאה של כל המשתמשים לאורך כל הסיבובים:

$$\mathbb{E}[R] = \mathbb{E}[\sum_{t=1}^{T} \sum_{n=1}^{N} X_n^{(t)}]$$

כאשר הרכיב ההסתברותי נובע מכך שתגובות הלקוחות X הינם משתנים מקריים.

דגשים חשובים

המטריצה P הינה כללית ולאו דווקא יש קשר בין הכניסות השונות בה. למרות זאת, לא תוכלו להתייחס לכל משתמש בנפרד בעקבות מגבלת התקציב על כלל ההמלצות.

קבצים מצורפים

לצורך המשימה, תקבלו שלושה קבצים:

1. הקובץ Recommender.py המכיל את המחלקה Recommender אותה עליכם להשלים:

```python
class Recommender:
    def __init__(self, n_weeks: int, n_users: int, prices: np.array, budget: int):
        self.n_rounds = n_weeks
        self.n_users = n_users
        self.item_prices = prices
        self.budget = budget

    def recommend(self) -> np.array:
        pass

    def update(self, results: np.array):
        pass
```

- הבנאי **__init__** מקבל כקלט את המתואר המשימה: מספר השבועות n_weeks, מספר המשתמשים n_users, מחירי ההפקה prices ומגבלת התקציב budget. המחירים ומגבלת התקציב הם מספרים שלמים (מטיפוס int). כמובן שאתם יכולים להוסיף עוד פונקציונליות מעבר לשמירת הקלט.
- הפעולה **recommend** נקראת בכל תחילת שבוע, ומחזירה את המלצות המערכת. על ההמלצות להיות מערך np.array באורך N, מטיפוס int. **דגש חשוב**: על ההמלצות להיות בין $0, \dots, K-1$
- הפעולה **update** נקראת בסוף כל שבוע, ומקבלת כקלט את תגובות הלקוחות $X_1^{(t)}, \dots, X_N^{(t)}$ בתור מערך results.

2. הקובץ simulation.py מכיל את המחלקה Simulation:

```python
class Simulation():
    def __init__(self, P: np.array, prices: np.array, budget, n_weeks: int):
        self.P = P.copy()
        self.item_prices = prices
        self.budget = budget
        self.n_weeks = n_weeks

    def _validate_recommendation(self, recommendation): ...

    def simulate(self) -> int: ...
```

- הבנאי **__init__** מקבל את המטריצה P, את מחירי ההפקה prices, את מגבלת התקציב budget, ואת מספר השבועות n_weeks.
- הפעולה **simulate** מבצעת סימולציה יחידה של מערכת ההמלצה, ומחזירה את התועלת שהתקבלה:

$$R = \sum_{t=1}^{T} \sum_{n=1}^{N} X_n^{(t)}$$

שימו לב: הפעולה **_validate_recommendation** בודקת את תקינות ההמלצות, הן מבחינת פורמט (טיפוס וצורה (shape)) והן מבחינת מגבלת התקציב. במקרה של שגיאה, העזרו בהדפסות שלה.

3. הקובץ test.py מכיל שלושה מקרים עליהם יבדקו ביצועי מערכת ההמלצה שלכם.

המקרים שמורים בפורמט הבא:

```
test_1 = {
    'P': np.array([
        [0.2, 0.6, 0.3, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.0],
        [0.1, 0.8, 0.5, 0.1, 0.0, 0.0, 0.0, 0.2, 0.0, 0.1],
        [0.0, 0.7, 0.2, 0.3, 0.0, 0.2, 0.1, 0.0, 0.0, 0.1],
        [0.1, 0.8, 0.2, 0.2, 0.2, 0.0, 0.0, 0.1, 0.0, 0.0],
        [0.0, 0.7, 0.1, 0.3, 0.0, 0.1, 0.0, 0.0, 0.4, 0.0],
        [0.0, 0.9, 0.2, 0.2, 0.2, 0.0, 0.1, 0.2, 0.0, 0.3],
        [0.1, 0.5, 0.2, 0.3, 0.0, 0.0, 0.1, 0.0, 0.1, 0.2]]),
    'item_prices': np.array([10, 10, 10, 10, 10, 10, 10, 10, 10, 10]),
    'budget': 20,
    'n_weeks': 1500
}
```

בסוף הקובץ מוגדרים:

```
tests = [test_1, test_2, test_3]
required_results = [7250, 2250, 4100]
```

כאשר ב-required_results שמורים ערכי הסף אותם אתם נדרשים לעבור (כפי שמוסבר בניקוד).

מגבלות זמני ריצה

לחברה ספוטיקסט זמן מוגבל, ולכן על הקוד שלכם במחלקה Recommender לרוץ תחת מגבלת זמן.

על כל הריצה (ביצוע יחיד של __init__ וביצוע של recommend ו-update בכל סיבוב) לקחת פחות מ-2 דקות. אם הריצה תמשך יותר שתי דקות, הרווח יתחשב רק בסכום הרווחים של הסיבובים שהסתיימו בשתי הדקות הראשונות. לדוגמה, אם נגמר הזמן באמצע הסיבוב החמישי, יוחזר הרווח של ארבעת הסיבובים הראשונים.

ניקוד

מעבר של ערכי הסף עבור המקרים בקובץ test.py מזכה ב-30 נקודות. המערכות שלכן יבדקו על קלטים נוספים, באופן תחרותי. החלק התחרתי יהיה שווה מעל ל-20 נקודות (100=50+30+20).

עבור החלק שאינו תחרותי, פלט לא חוקי/חריגה מזמני הריצה תיתן ערך 0 בסימולציה הנוכחית, אך עדיין יאפשר לקבל ניקוד מלא אם ביצועי המערכת יהיה מספיק טובים בשאר הסיבובים.

חריגות כאלה בחלק התחרותי יגררו ניקוד 0 בכל החלק.

ספריות מותרות

מלבד הספרייה הסטנדרטית של Python, הספריות החיצוניות המותרות לשימוש הן numpy ו-scipy בלבד (השימוש ב-numpy חובה). אין להשתמש בספריות לחישובים מקביליים דוגמה threading, multiprocessing, ספריות נוספות שאינן רלוונטיות דוגמת os ועוד.

הוראות הגשה

שימו לב - הגשה לא תקינה תכלול הורדת ציון!

עליכם להגיש תיקיית ZIP בשם HW2_PART2_ID1_ID2.zip המכילה את הקבצים הבאים בלבד

- HW2_PART2_ID1_ID2.py – קובץ Python יחיד בשם המכיל את המימוש שלכם למחלקה Recommender.
  כל הקוד שלכם שתצטרכו כדי להריץ את המחלקה צריך להיות בקובץ הזה.
- HW2_PART2_ID1_ID2.pdf – קובץ PDF עם תיאור קצר (עד עמוד) המתאר את הגישה שלכם לפתרון.