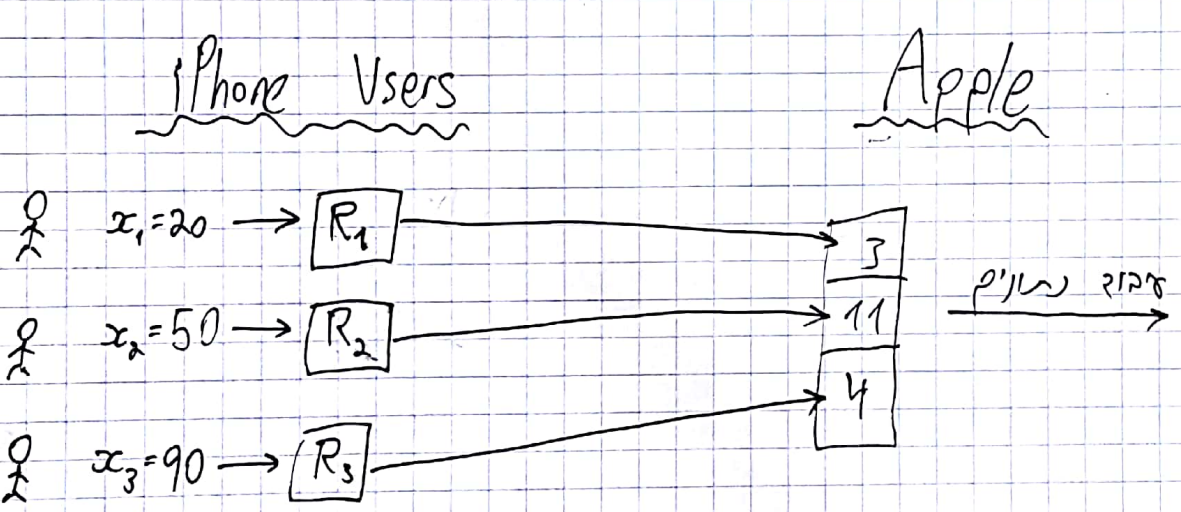
Local Differential Privacy

שאלה: ענית שחברה מסוימת (למשל Apple) רוצה לדעת את ההרגלים של הלקוחות שלה ולכן מדגם מסוימת מהם מאזכר בהרגלים רגישים (מבחינת פרטיות) אך יכול להיות שהלקוחות לא רוצים לשלוח לחברה את ההרגלים שלהם.

איך החברה יכולה לדעת את ההרגלים תוך כדי שהיא מבטיחה ללקוחות שהיא לא תוסיף את ההרגלים שלהם?



* ישנם n משתמשים, כאשר כל משתמש i מחזיק קלט x_i. נסמן $S=(x_1, x_2, \ldots, x_n)$. נתמקד על S בהקשר "קלה ביותר מבוקרת". אני חושב על S בהקשר "קלה ביותר מבוקרת" כי הוא לא נמצא במקום אחד, אלא כל משתמש מחזיק את הנתון שלו אצלו.
* יש שרת שמעוניין לדעת את אופן ההרגלים S
* המשתמשים לא שולחים את הקלטים שלהם לשרת. ~~[redacted]~~ כל משתמש מפעיל (באופן מקומי) אלגוריתם פרטי על הקלט שלו, ושולח לשרת את תוצאת החישוב.

הגדרה: אלגוריתם $R: X \to Y$ נקרא ε-randomizer אם לכל זוג קלטים $x, x' \in X$ ולכל פלט $y \in Y$ מתקיים

$$\Pr[R(x)=y] \le e^{\varepsilon} \cdot \Pr[R(x')=y]$$

כלומר לא ניתן להבדיל האם הפלט הגיע מ-x או מ-x', תוך מתן שקלול את שלנו הקבוע e^{ε}.

הגדרה: אלגוריתם $M: X^n \to W$ מקיים ε-LDP אם הוא עונה לכל אחד מהקלטים של היורד פשוט אותו, דרך ε-randomizer כלשהו.

הערה: באופן יותר כללי, ניתן לבנות k פונקציות $R_1, \dots, R_k$ מסוים, כך שהרכבתם $\varepsilon = \varepsilon_k + \dots + \varepsilon_1$ היא ε-randomizer.

* לתשומת לבכם ההבדל בין ההגדרה הראשונה מעלה הוא

מנגנונים נפוצים ל-LDP

* מפשר ללמוד על כלל האוכלוסייה תוך הגנה על פרטיות הנקודה מסוים למשתמשים
* רוב השאילתות נשמרים אינם השרת חושף את כל התשובות שכן לכן בין משתנה, זו אינם יש בריבוי לשרת אבל הקודם הנלווים לא ניתן בחינם, כי יש לנו הרבה פעמים הגדרה (כל משתמש מוסף רעש) ואותנו ניפוים לרוב אינם מהשאלות סמן לה.

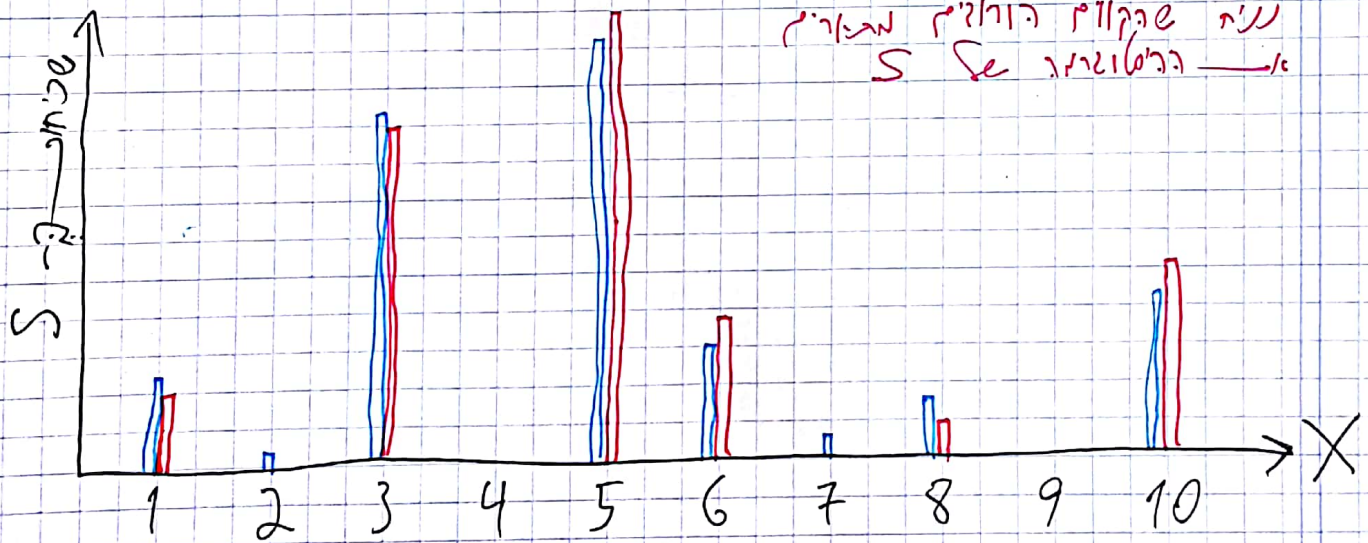
הגדרת בעיה: שערוך לפי דגימה קלה

- נניח n מדגמים, כאשר כל מדגם $x_i \in X$.
  סימון $S = (x_1, x_2, \ldots, x_n)$
- **הגדרה:** לכל קלט אפשרי $x \in X$ נרצה להעריך את השכיחות של x ב- S, המסומנת ב- $f_S(x)$

  לזאת נכל $x \in X$ אומר מספר המופעים של x ב- S

דוגמה:

עמודת שכיחות הדוגמים מייצגים את ההתפלגות של S



אנחנו מחפשים לקבל הערכה $\hat{f}_S(x)$ לכל $x \in X$ כך שהשגיאה

$$\max_{x \in X} \left| f_S(x) - \hat{f}_S(x) \right|$$

כלומר הקטנה ככל האפשר.

מוטיבציה:

- חברות גדולות — כמו Google ו Apple — משתמשות בכמות אדירה של מידע כדי לשפר את השירותים שלהן
- סיבוכיות זמן — מדגם גדול מדי יגרום לכך שהאלגוריתם יהיה איטי מדי

④ שאלת בונוס: אילו הזוגות X הוגי בקול מאוד, ס"ל
שלכל $x \in X$ ניתן לחשב $\hat{f}_S(x)$ בזמן מהיר.
ובמילים פשוטות של Apple ואיזה הזוגות X הוא בקול מאוד.

איך נעקוף את המשפט?

דרך 1: נחפש אלגוריתם אשר מוציא הריבה בקלט מעט
יחסית ומאפשר לחשב $\hat{f}_S(x)$ לכל $x \in X$.
אלגוריתם כזה נקרא Frequency Oracle.
כלומר אולי כזה לוז כותב באופן מפורש את הקירוב שהוא
חשב, אלא במוצא המדב הוא מין פ' קסם שלום אינו מציין
לך איבר x היא מעריך פ' הערכה $\hat{f}_S(x)$.

דרך 2: נחפש אלגוריתם אשר מוציא הריבה בקלט רשימה
קצרה של איברים $X \subseteq T$ (נעיר בהמשך $|X| \leq n$) ובמוצא
פולט הערכה $\hat{f}_S(x)$ לכל $x \in T$. המוסכמה היא
שמעריכים $\hat{f}_S(x) = 0$ לכל $x \notin T$.
* לחצור לציור הטוב ביותר שיכסה את כל אחד שם.
אלגוריתם כזה נקרין אלגוריתם למציאת heavy-hitters.

הערות:

① בשאלת האפשרויות אנו מעריכים את השגיאה של הלוע כ-
$$\max_{x \in X} \left| f_S(x) - \hat{f}_S(x) \right|$$

② כל אלגי למציאת heavy-hitters הינו בפרט Frequency Oracle
לכן אפשרות 1 היא (אולי) קלה יותר להשגה.

③ למה שיקול עין הריבה ~~[illegible]~~ באפשרויות הינו
כי אם יש ל' Freq. Oracle אולם לשאול יותר אחד על כל איבר
הזוגות עד שאנחנו ← הבנתי.

תוכנית הלימוד הבאה:

1. נראה הוכחה אחידה ל- heavy-hitters עבור כל Freq. Oracle (הוכחה משותפת עליהם). כלומר נראה כיצד בהינתן אלג' Freq. Oracle נתון נבנה אלג' למציאת heavy-hitters.
2. נראה כיצד לבנות Frequency Oracle תחת LDP.

חלק 1: Frequency Oracle $\Rightarrow$ Heavy-hitters

משפט 1: אם קיים Frequency Oracle עם שגיאה α תחת ε-LDP וסיבוכיות $O(z)$ אז קיים אלג' למציאת heavy-hitters עם שגיאה α תחת ε-LDP עם $(n+z)$ אותו סדר גודל, וזיכרון, וזמן ריצה.

אלגוריתם ההוכחה של משפט זה יהיה:

משפט 2: אם קיים Frequency Oracle כלשהו עם שגיאה α תחת ε-LDP אז קיים אלג' כלשהו למציאת heavy-hitters עם שגיאה 2α תחת $\varepsilon \cdot \log|X|$-LDP.

הוכחה (נתאר אלגוריתם למשפט 2):

סימונים: (1) נסמן ב- $\mathcal{O}$ את אלג' ה- Frequency Oracle הנתון תחת ε-LDP ושגיאה α

(2) ישנם n משתמשים כאשר לכל $i \in [n]$, למשתמש i מתאים קלט $x_i \in X$. נסמן $S = (x_1, \ldots, x_n)$

(3) תהי $h: X \to [T]$ פונקציית hash (בדומה לבלום) המתאימה באופן אקראי לכל $x \in X$ ערך T.

הגדרה מפתח: אין התנגשויות בפונקציית ה-hash h עבור S, אם לכל $x_i \neq x_j \in S$ מתקיים $h(x_i) \neq h(x_j)$.

כלומר יש לנו פתרון Σ^2 ל-$(1^{*}x[l]-1, t^{*})$ מופיע בדיוק 0 פעמים ב-S_l.

מצד שני, מכיוון שהנחנו שאין התנגשויות ב-h, אז $(x^{*}[l], t^{*})$ מופיע לכל היותר Σ^2 פעמים ב-S_l (עבור l).

- מכיוון ש-x^{*} מופיע ב-S לפחות Σ^2 פעמים, ולכן $(x^{*}[l], t^{*})$ מופיע לפחות Σ^2 פעמים ב-S_l לכל l, מבוצר S ו"ברור" מתוך $\log|X|$ הבדידים מבוצרים S_l כשורה אלו ב-S_l בדומה. כלומר ההתלבטות היא שמא h יצור לנו, כי משהו i יוכל להשפיע על x_i.

- עבור כל $l \in [\log|X|]$ נגדיר $S_l = \left(h(x_i), x_i[l]\right)_{i\in[n]}$

- נקח את $x^{*} \in X$ המופיע לפחות Σ^2 פעמים ב-S. נסמן $t^{*} = h(x^{*})$.

- נקח את $x^{*} \in X$ המופיע לפחות Σ^2 פעמים ב-S. כלומר $\Sigma^2 \le |\{x : h(x) = t^{*}\}|$. המטרה שלנו היא לבחור את x^{*}.

הערה: המטרה שלנו היא להשתמש ב-$\mathcal{O}$ על מנת לבחור את כל האיברים שמופיעים לפחות Σ^2 פעמים ב-S, מבלי לבצע חיפוש ממצה על X.

ברעיון: במקום לבצע חיפוש ממצה על X הינו רוצים לבצע חיפוש ממצה על $[T]$ (שהוא קטן משמעותית). אבל איך?

רמז לפתרון: אנו אפשר את $\mathcal{O}$ כמה פעמים שנרצה (כלומר פי אפשר יהיו $\mathcal{O}$ שיהיו על S)

לכן עבור x^* כלשהו, ה-j-ה בוקטור $(1,t^*),(0,t^*)$ יהיה שונה מאפס אם ורק אם $x^*_j=1$. לכן ניתן לשחזר את x^* בעזרת Freq. Oracle אחד.

האלגוריתם:

לכל $t\in[T]$ נחשב אומד $\hat{x}^{(t)}$ באופן הבא:

- לכל $j\in[\lfloor\log x\rfloor]$ נקבע את $(S)0$ המתאים ל-j-ה בוקטור $(1,t),(0,t)$ ונחשב $\hat{f}_S(1,t),\hat{f}_S(0,t)$

$$\hat{x}^{(t)}[j]\leftarrow \underset{g\in\{0,1\}}{\operatorname{argmax}}\left\{\hat{f}_S(g,t)\right\}$$

לבסוף, נחזיר את $\hat{x}^{(t^*)}$ כאשר t^* הוא ה-t הראשון שעבורו $\hat{x}^{(t)}=\hat{x}^{(t-1)}$. נטען שבמקרה זה $\hat{x}^{(t^*)}=x^*$.

כלומר, האלגוריתם הנ"ל מחזיר את x הראשון שעבורו T אומדים, $\hat{x}$, מסכימים ביניהם: $\exists x\in\{0,1\}^d$ כך ש-$\forall t\geq x$ מתקיים $\hat{x}^{(t)}=x$.

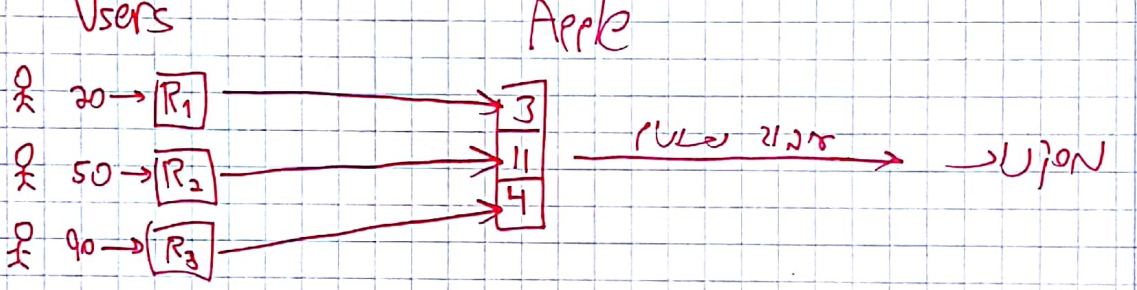
טענה: כל שהוא המשתנה t^* שהוגדר לעיל מקיים $t^*\geq T$ ולכן עבור t^* זה מתקיים $\hat{x}^{(t^*)}=x^*$.

הוכחה: נשים לב כי $\hat{x}^{(t)}$ הוא פונקציה של $\hat{f}_S(0,t),\hat{f}_S(1,t)$ לכל S ולכן של $(0,t),(1,t)$ לכל S. לכן עבור $t\geq T$ כלשהו, $\hat{x}^{(t)}$ הוא פונקציה של $(0,t^*),(1,t^*

חלק ב': בניית Freq.Oracle משתמר ε-LDP

בשיעור הקודם הגדרנו את מודל LDP, ושם ראינו את הפרוטוקול הפשוט ביותר שמשתמר את הפרטיות הנדרשת. אלא שבפועל הפרוטוקול הזה אינו שימושי כלל, משום שהוא מוסיף כמות עצומה של רעש. בשיעור הזה נראה כיצד ניתן לבנות Freq.Oracle שמשתמר את הפרטיות הנדרשת אך מוסיף כמות רעש סבירה.

שאלה מנחה: איך ניתן לבנות אפליקציה כמו Apple שמשתמרת את פרטיות המשתמשים?



תזכורת: נניח $X=\{\pm 1\}$. נתון Freq.Oracle שמתאר נתונים בינאריים

- פרמטרים: מקבל $S=(x_1,\dots,x_n)$ כאשר $x_i \in \{\pm 1\}$ שמתאים למשתמש i.
- מטרה: להעריך את מספר ה-1-ים ב-S (מספרים: $\sum_i x_i$).

נשתמש ב-Randomized Response:

Randomized Response

קלט: $x \in \{\pm 1\}$

פלט: מחזיר x בהסתברות $e^\varepsilon/(e^\varepsilon+1) = \frac{1}{2}+\frac{\varepsilon}{4}$

ומחזיר $-x$ בהסתברות $1/(e^\varepsilon+1) \approx \frac{1}{2}-\frac{\varepsilon}{4}$

מ"ו: אלג' $R(\cdot)$ מהצורה הבאה הוא ε-randomizer

הפרוטוקול: כל משתמש i מקבל הודעה $y_i \leftarrow R(x_i)$.

שרת מחשב $\frac{3n}{\varepsilon}\left(\sum_{i\in[n]} y_i + \frac{n}{3}\right)$

⑨

$$\mathbb{E}\left[\sum_{i\in[n]} y_i\right] = \sum_{i:\,x_i=1} \mathbb{E}[y_i] + \sum_{i:\,x_i=-1} \mathbb{E}[y_i]$$

נפרוס את השאר:

$$\mathbb{E}[y_i] = (-1)\cdot\left(\frac{1}{2}+\frac{\varepsilon}{2}\right) + 1\cdot\left(\frac{1}{2}-\frac{\varepsilon}{2}\right) = -\varepsilon$$

$$= (-1)\cdot f_S(-1)\cdot(\quad) + f_S(1)\cdot(\quad) =$$

$$= (1-f_S(1))\cdot(\quad) + f_S(1)\cdot(\quad) =$$

כלומר שתי השאלות שהוקלקו שלנו הן אכן

כי ניתן חסם מהסתברות על השאלה, שנקרא להשתמש בחסם הזה:

משפט: יהיו $X_1,\dots,X_n$ משתנים מקריים בלתי תלויים בקטע $[a,b]$.

אז, לכל $t\geq 0$:

$$\Pr\left[\left|\sum X_i - \mathbb{E}\left[\sum X_i\right]\right| \geq t\right] \leq 2\cdot\exp\left(-\frac{2t^2}{n\cdot(b-a)^2}\right)$$

במקרה שלנו $a=-1$ $b=1$ ולכן עבור $\sum y_i$ שמתקיים

טענה: שווה ל $\sqrt{\ln(\frac{2}{\beta})\cdot 2n}$ והסיכוי לכך גדול מ β.

ולכן בהסתברות $\geq (\beta-1)$ שגיאת ניבוי לכל היותר $\sqrt{\ln(\frac{2}{\beta})\cdot\frac{1}{2n}}$

עכשיו התיאוריה. מתי אנחנו הכי מפחדים לבחור Freq.Oracle

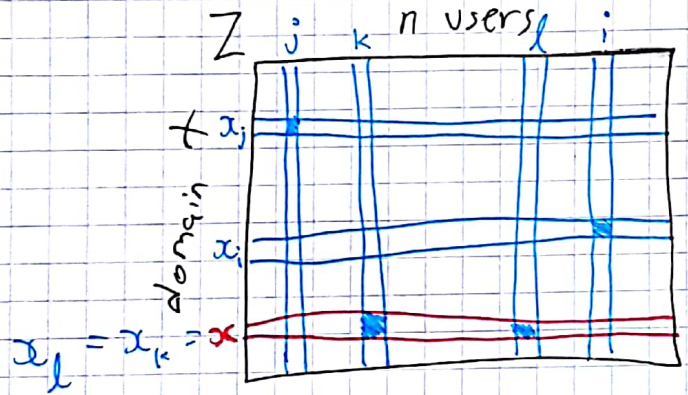
נזכר קודם X שנקרא:

קל"ה של Freq. Oracle

- קלט: מקורי $S=(x_1,\dots,x_n)$ כאשר משתמש i מחזיק $x_i \in X$
- מטרה: בנה $x \in X$ אותו הוגים להיות מעלים לחשב בזה כמה x מופיע ב-S. $\hat{f}_S(x)$! מס' המופעים של $f_S(x)$

סימון: יהי $Z \in \{\pm 1\}^{n \times k}$ מטריצה שבחרה באקראי באופן אחיד, וניניח כי Z ידוע לכולם.

בנוסף:



(*) באלג' בא נשתמש (לאור הגדרה ה randomizer-ג):
- משתמש i מחזיק קלט x_i.
- 1) שולח ל-i' $Z[i',x_i]$ שמחזיר.
- 2) שולח לשרת response bit עבור $Z[i',x_i]$, כלומר:
  שולח $y_i = Z[i',x_i]$ בהסתב' $\frac{1}{2}+\frac{\varepsilon}{2}$
  ושולח $y_i = -Z[i',x_i]$ בהסתב' $\frac{1}{2}-\frac{\varepsilon}{2}$

ברגע שמי בחיזוק שמשתמש, הלך הבא מוכן על ידי randomizer-ג

(*) באלג' בא נשרת:
בהינתן שאילתא $x \in X$, חשב ומחזיר

$$\hat{f}_S(x) = \frac{1}{\frac{1}{2}\varepsilon} \sum_{\text{משתמש } i} y_i \cdot Z[i', x]$$

(11)

$$\mathbb{E}\left[\sum_{i\in[n]} y_i \cdot Z[x_i]\right] = \sum_{i:\, x_i = x} \mathbb{E}\left[y_i \cdot Z[x_i]\right] + \sum_{i:\, x_i \neq x} \mathbb{E}\left[y_i \cdot Z[x_i]\right]$$

$$= \mathbb{E}Z \cdot f_S(x)$$

כלומר, שוב קיבלנו הטיה בגודל שלט הוא אלפ.

כאן קורן, $\sum y_i Z[x_i]$ הוא סכום של n גדלים בי"ת.

נזכיר שקלים שגודלו $(\beta\cdot t)$ הכולים קרוב לוודאי

שלא יהיו כאלו $\sqrt{2n\cdot\ln\left(\frac{2}{\beta}\right)}$, ונגדיר כך ערך שיטה

$\frac{\mathbb{E}Z}{t}\sqrt{2n\cdot\ln\left(\frac{2}{\beta}\right)}$ ביחס לזה