(23) 6/12/09

סטה וחישוב

* הודעה: ביום חמישי יש כנס של משחקים שמאוד קשורים לקורס. סטודנטים ששוקלים להמשיך לחקר בתחום מוזמנים מאוד להשתתף בכנס. מי שמעוניין שישלח מייל למיכל או לנועם.

Israeli Seminar on CGT

* הודעה - מי שלא נרשם לסיכום שיעורים ירוץ לבקש לפרסם פתרון בוס לתרגיל כלשהו.

משחקי סכום אפס משני שחקנים

המשחק מתואר ע"י מטריצה $A=\{a_{ij}\}$ אשר מתארת את התשלום ששחקן ה-min (עמודות) משלם לשחקן ה-max (שורות).

שחקן ה-max רוצה למקסם את התשלום שלו, כלומר הבעיה שהוא רוצה לפתור היא

$$\max_x \min_j \sum x_i a_{ij}$$
$$\text{s.t.}\quad x_i \geq 0 \;\; \forall i$$
$$\sum x_i = 1$$

זאת תכנית כמעט לינארית. תכנית לינארית שקולה היא

$$(1)\qquad \begin{aligned} &\max\; c \\ &c \leq \sum x_i a_{ij} \;\; \forall j \\ &x_i \geq 0 \;\; \forall i \\ &\sum x_i = 1 \end{aligned}$$

קל להשתכנע שזאת אכן תכנית שקולה.

מאידך, שחקן ה-min דווקא רוצה למזער את ההפסד שלו. כלומר הוא רוצה לפתור את התכנית

$$\min_{y} \max_{i} \sum_{j} a_{ij} y_j$$
$$\text{s.t} \quad y_j \ge 0 \ \forall j$$
$$\sum y_j = 1$$

שזה הינו נראה ליניארית

אבל יש לה תכנית ליניארית שקולה

(3)
$$\min_{y} d$$
$$\text{s.t} \quad d \ge \sum a_{ij} y_j \ \forall i$$
$$y_j \ge 0 \ \forall j$$
$$\sum y_j = 1$$

נתבונן בתכנית

(2)
$$\min \sum x_i$$
$$\text{s.t} \quad x_i \ge 0 \ \forall i$$
$$\sum x_i a_{ij} \ge 1 \ \forall j$$

<u>טענה:</u> האופטימום של (1) שווה בדיוק להופכי של האופטימום של (2).

<u>הוכחה:</u> אם $X=(X_1,\dots,X_m)$ פתרון אפשרי ל-(1) ונותן איזה ערך c.
אז $\frac{X}{c}=(\frac{X_1}{c},\dots,\frac{X_m}{c})$ הוא פתרון אפשרי ל-(2)
והוא נותן ערך $\sum \frac{X_i}{c} = \frac{1}{c}$.
לכן $OPT(2) \le \frac{1}{OPT(1)}$

מצד שני, אם x' פתרון אפשרי ל-(2) ונותן ערך $\frac{1}{c}$ אז
$c \cdot x'$ פתרון אפשרי ל-(1) שנותן ערך c. לכן
$OPT(1) \ge \frac{1}{OPT(2)}$ $\Leftarrow$ יש שוויון.

באופן דומה עבור שחקן ה-min נתבונן בתכנית

(4)
$$\max \sum y_j$$
$$\text{s.t}$$
$$y_j \ge 0 \ \forall j$$
$$\sum_j a_{ij} y_j \le 1$$

אפשר להניח בה"כ ש- $a_{ij} > 0$ כי אחרת פשוט נוסיף קבוע למטריצה ושום דבר לא משתנה.

ושם מתקיים $OPT(3) = OPT(4)$

אבל תכניות (4) ו-(2) הן דואליות ולכן

$$OPT(4) = V_{max} = V_{min} = OPT(2)$$

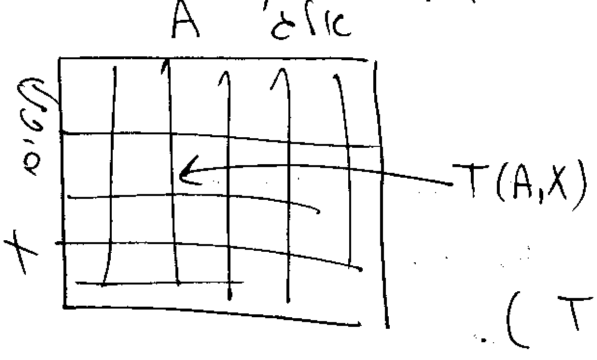
$\Leftarrow$ $OPT(1) = OPT(3)$ (כי הם ההופכיים של (2) ו-(4))

$$\Leftarrow \quad \max_x \min_j \sum_i x_i a_{ij} \ \ \text{s.t. } x_i \ge 0\ \forall i,\ \sum x_i = 1 \quad = \quad \min_y \max_i \sum_j a_{ij} y_j \ \ \text{s.t. } y_j \ge 0\ \forall j,\ \sum y_j = 1$$

ובזאת הוכחנו את משפט המינמקס. הוא משפט מאוד חשוב והוא למעשה אומר שברגע שיש לנו שני שחקנים, אין חשיבות למי שמשחק קודם.
מעבר לזה, השחקנים יכולים לחשב את האסטרטגיה האופטימלית עבורם, ובעזרת תכנון לינארי אפשר אפילו לעשות את זה בזמן פולינומיאלי. האסטרטגיות האלה הן גם ש"מ נאש מעורב.

עכשיו נדון באפליקציות טריוויאליות של המשפט הזה.

נסתכל על משחק סכום אפס: יש איזו בעיה לפתור ויש m מיני אלג' דטרמיניסטיים שפותרים אותה. מהיותו אלג' A וקלט X נסמן ב- $T(A,X)$ את זמן הריצה של A על X.



אסט' A של שחקן האלגוריתמים מבטיחה תוצאה $\le T$ כאשר T הוא זמן הריצה של A ב- worst case (כלומר $T = \max_X T(A,X)$).

שחקן האלגוריתמים רוצה למזער את זמן הריצה. היריב, שחקן הקלטים יכול לתת קלטים שקשה לעבוד עליהם - היריב מנסה לתת לנו קלטים גרועים. זה מודל טוב כי מתאר אנשי מדעי המחשב מה שמעניין אותנו הוא ה- worst case.

אם היינו יודעים את ההתפלגות D של הקלטים האפשריים שבהן האלגוריתם היה מוצא את התשובה הטובה ביותר ל-D.
התשובה עלולה להיות מעורבת אבל היא לא חייבת. תמיד אפשר למצוא אסט' טהורה שממצערת את התוחלת.
אבל הצרה היא שאנחנו לא יודעים את D ובכלל אין התפלגות קבועה והיא משתנה עם הזמן.

- זמן הריצה של האלג' הטוב ביותר הוא

$$\min_A \max_X T(A,X)$$

- זמן הריצה הממוצע הטוב ביותר של התפלגות D של הקלטים הוא

$$\min_A T(A,D)$$

כאשר

$$T(A,D) = \mathbb{E}_{X\in D}\left[T(A,X)\right]$$

- זמן הריצה הממוצע של האלג' הטוב ביותר של ההתפלגות הגרועה ביותר הוא

$$\max_D \min_A T(A,D) =$$

$$\overset{\text{משפט המינמקס}}{=} \min_{\underset{\text{התפלגות של } A}{R}} \max_D T(A,D)$$

אז קיימת התפלגות של האלג' הדטרמיניסטיים שלכל קלט מבטיחה זמן ריצה לכל היותר $\max_D \min_A T(A,D)$,
אז אפשר לחשוב על אלג' הסתברותי כהתפלגות של אלג' דטרמיניסטים.

<u>הלמה של Yao</u>: זמן הריצה הממוצע של האלג' הטוב ביותר על ההתפלגות הגרועה ביותר שווה לזמן הריצה הממוצע של האלג' ההסתברותי הטוב ביותר על הקלט הרע ביותר.

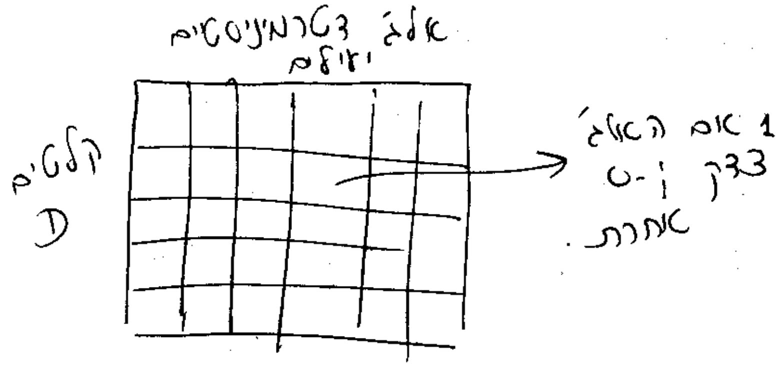
25

הערה: הכיוון ($\geq$) הוא טריוויאלי. זמן הריצה של האלג' הטוב ביותר הוא כמובן חסם תחתון למה שאלג' הסתברותי יכול לעשות. כי זמן הריצה של אלג' הסתברותי הוא ממוצע של זמני הריצה של אלג' דטרמיניסטיים שמתחתיו.
הכיוון השני נובע ממשפט המינימקס (למעשה גם הראשון אבל הוא לא קשה גם כבר). ☺

הלמה הזאת חשובה כי היא נותנת לנו טכניקה למצוא חסמים תחתונים לאלגוריתמים הסתברותיים.

עד כה דיברנו על אלגוריתמים שמעולם לא טועים והזמן היה זמן הריצה שלהם.

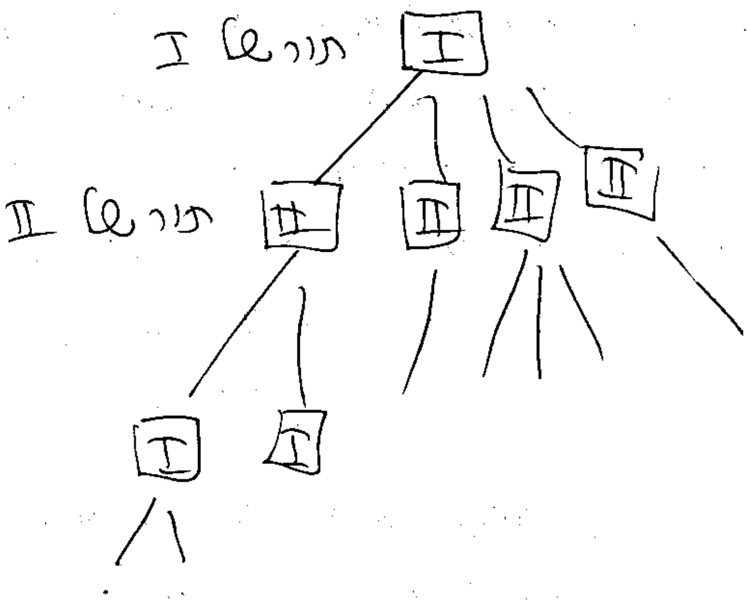
נסתכל על המשחק הבא:



אם אלג' הוא פשוט נכון תמיד אז התמורה שלו היא כולה 1.
אם אלג' צודק על 90% מהקלטים אז בממוצע שלו יש 90% 1.
אלג' הסתברותי צודק ב-90% על קלט אם בשורה המתאימה יש 90% 1 (התוחלות הן ממוצע לפי ההתפלגות)
ואפשר גם לדבר על אלג' שצודק ב-90% של D הקלטים - על D קלט זה צריך להיות 90%. אז גם לאלג' שלא תמיד צודקים אפשר לטעון משהו מאוד דומה ללמה של Yao.

שיערוך של עצי משחק Game-tree evaluation

משחק אפשר לתאר באמצעות עץ. בכל תור לשחקן יש אוסף סופי של צעדים שאפשר לעשות.



העץ מתאר את אוסף ההתפתחויות האפשריות - טומן את
חוקי המשחק - מה יכול להיות בכל שלב. כל קודקוד מתאר
היסטוריה מסויימת והצלעות שיוצאות ממנו אומרות מה יכול
לקרות בהינתן ההיסטוריה הזאת.
בעלים מתואר מה קורה בסוף המשחק. למשל אם זה משחק סכום
אפס בעלים אפשר לרשום רק - התשלום לשחקן I.
ההצגה הזאת נקראת הצגה מורחבת (extensive) של
משחק.
בהצגה כזאת אסט' של שחקן היא בחירה של פעולה לכל קודקוד
שבו זה התור של השחקן הזה. כך, אפשר להפוך את
המשחק לצורה נורמלית (מטריצה).
כדי לפתור משחק כזה אפשר לעשות אינדוקציה לאחור.
בצעד האחרון ממש קל להחליט מה לעשות. שחקן I
פשוט ילך לעלה עם הערך הכי גבוה. משם אפשר
להחליט מה צריך לשחקן II לעשות. ככה אפשר לפעפע
למעלה ולקבל את הערך.

<u>קלט</u>: עץ משחק
<u>פלט</u>: הערך של המשחק
<u>אלג'</u>: מפעפעים למעלה.

```
eval(v)
  if v is
      return lv
  if v is min-node (II)
      current min = ∞
      for all children u do
          X = eval(u)
          curren min = min(X, current min)
      return current_min
  else //(v is max node)
      current max = -∞
      for all children u do
          X = eval(u)
          current max = max(X, current max)
      return current_max
```

האלג' הזה לינארי בגודל העץ. מפה יכול לעניין אותי
כמה עלים חייבים לקרוא כדי לשערך. האלג' הזה
קורא את כולם אבל אולי לא חייבים...
שבוע הבא נראה שבאופן כללי * חייבים לקרוא הכל אבל
באופן הסתברותי אפשר לחסוך...