



קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

תכנון דינאמי - בעזרת טבלה

Value Iteration



ריצארד בלמן 1920 - 1984



תוכן השיעור

- חזרה על האלגוריתמים שלמדנו:

- .Policy Evaluation

- .Policy Improvement

- .Policy Iteration

- שיפור ויעול האלגוריתם – Value Iteration:

- ייעול מספר האיטרציות שיביא להתכנסות האלגוריתם.

- שימוש ב Value Table ללא צורך ב Policy Table.

- הדגמה של האלגוריתם באמצעות "פאזל המספרים".

משוואות בלמן בסביבה דטרמיניסטית

$$\Pi(s) = a \quad | \quad P(s) = a$$
$$Model(s, a) = s', r$$

• משוואה הערך $V(s)$ בסביבה דטרמיניסטית היא:

$$V_p(s) = r + \gamma V_p(s')$$

• משוואת $Q(s, a)$ בסביבה דטרמיניסטית:

$$Q_p(s, a) = r + \gamma Q_p(s', a')$$

חישוב פונקציית הערך למדיניות נתונה

Policy Evaluation

$$V_p(s) = r + \gamma V_p(s')$$

האלגוריתם של בלמן מציע לבצע חישוב חוזר ונשנה של המשוואה על הטבלה עד אשר לא יהיו יותר שינויים בטבלה (או שהשינויים יהיו קטנים מאוד):

	🤖	-1	
			+1

↓	↓	↓	↓
↓	↓	█	↓
↓	↓	↓	↓
↓	↓	→	█

0	0	0	0
0	0	█	0
0	0	0	0
0	0	0	█

0	0	-1	0
0	0	█	0
0	0	0	1
0	0	1	█

0	0	-1	0
0	0	█	0.9
0	0	0.9	1
0	0	1	█

0	0	-1	0.81
0	0	█	0.9
0	0	0.9	1
0	0	1	█

שיפור פונקציית המדיניות Policy Improvement

- אם נתונה לנו פונקציית הערך של מדיניות מסוימת אנחנו יכולים לשפר את המדיניות באמצעות פונקציית הערך לפי הנוסחה הבאה:

$$P'(s) = \max_a (r + \gamma V(s')) \quad \text{או} \quad P(s) = \operatorname{argmax}_a (r + \gamma Q(s, a))$$

- במילים אחרות, לכל מצב s אנחנו בוחרים את הפעולה שתביא אותנו ל $v(s')$ המירבי.

- Policy improvement מצריך איטרציה אחת בלבד.

	🤖	-1	
			+1

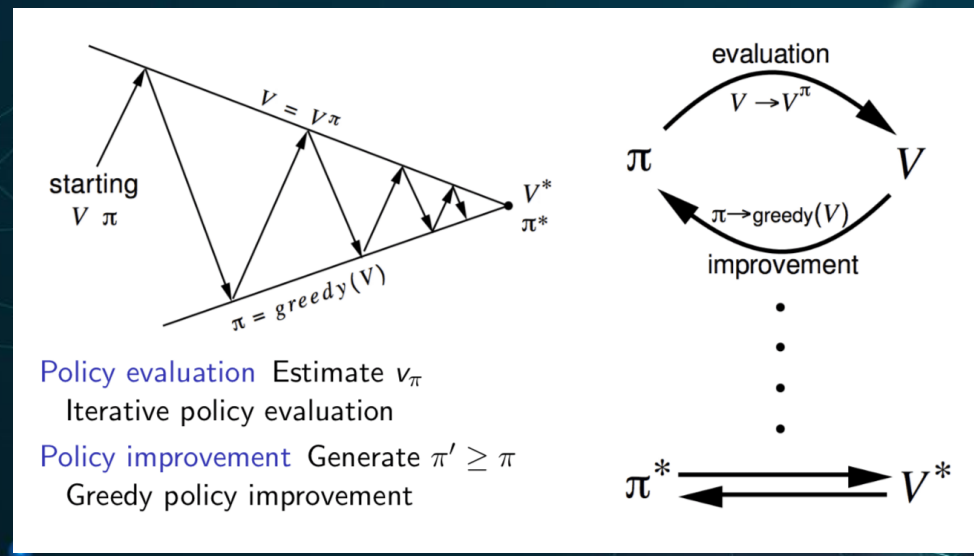
↓	↓	↓	↓
↓	↓	█	↓
↓	↓	↓	↓
↓	↓	→	█

0	0	-1	0.81
0	0	0	0.9
0	0	0.9	1
0	0	1	0

↓	↓	→	↓
↓	↓	█	↓
↓	→	↓	↓
↓	→	→	█

Policy Iteration

- אלגוריתם למציאת המדיניות המיטבית.
- אנו מבצעים לסירוגין:
 - Policy Evaluation – לעדכון טבלת הערכים באמצעות איטרציות חוזרות ונישנות.
 - Policy improvement – לעדכון הפולסי לפי טבלת הערכים החדשה.
- בלמן הוכיח כי בסופו של דבר אנחנו נתכנס למדיניות המיטבית.

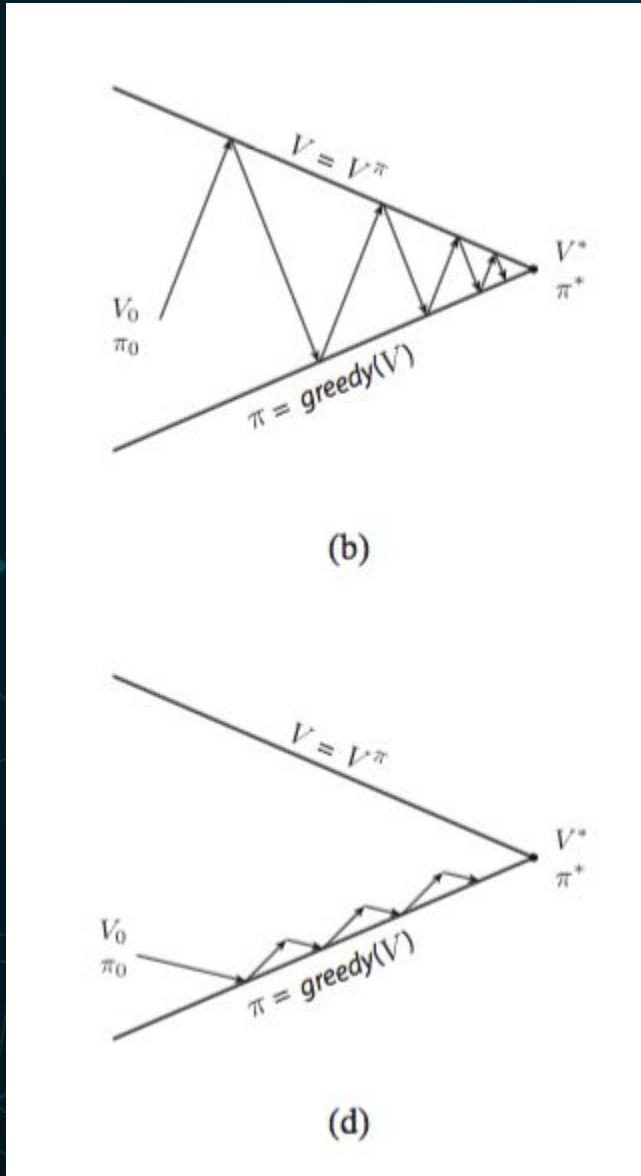


האם יש צורך בטבלת Policy

- באלגוריתמים שלנו השתמשנו בטבלה המחזיקה את הפעולות שיבוצעו על ידי הסוכן בכל מצב נתון $P(s) = \text{action}$, וזאת לצד טבלת ערכים.
- האם באמת יש צורך בטבלת מדיניות (פוליסי) ? התשובה היא לא !
- אנחנו יכולים להשתמש בטבלת הערכים לחישוב המדיניות בדומה לחישוב שעשינו ב Policy Improvement.
- בהינתן לנו מצב s ומודל ידוע נחשב לכל אחת מהפעולות האפשריות מהו המצב s' ומהו התגמול (Reward) שיתקבל.
- באמצעות טבלת הערכים נקבל את $V(s')$ ונחשב את $r + \gamma V(s')$
- נבחר את הפעולה שתביא לנו את הערך הגבוה ביותר.
- בדרך זו השתמשנו בטבלת הערכים לחישוב המדיניות שלנו.

0	0	0.729	0.81
0	0.729	0	0.9
0.729	0.81	0.9	1
0.81	0.9	1	0

ייעול האלגוריתם



• ניתן ליעל את האלגוריתם של Policy Iteration בשניים:


- נותר על עדכון טבלת המדיניות (Policy Improvement), שכן ברגע שיש לנו טבלת ערכים מעודכנת - אוטומטית יש לנו גם טבלת מדיניות חדשה.
- נשתמש בטבלת הערכים לחישוב הצעד הבא (המדיניות שלנו) תוך כדי עדכון טבלת המצבים.

• אנחנו לא מחכים לעדכון המדיניות לאחר סיום כל האיטרציות הנדרשות ל Policy Evaluation.

• לאחר איטרציה אחת אנחנו למעשה מעדכנים את המדיניות.



Value Iteration

			
		-1	
			+1

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

0	0	0	0
0	0	0	0
0	0	0	1
0	0	1	0

0	0	0	0
0	0	0	0.9
0	0	0.9	1
0	0.9	1	0

0	0	0	0.81
0	0	0	0.9
0	0.81	0.9	1
0.81	0.9	1	0

0	0	0.729	0.81
0	0.729	0	0.9
0.729	0.81	0.9	1
0.81	0.9	1	0

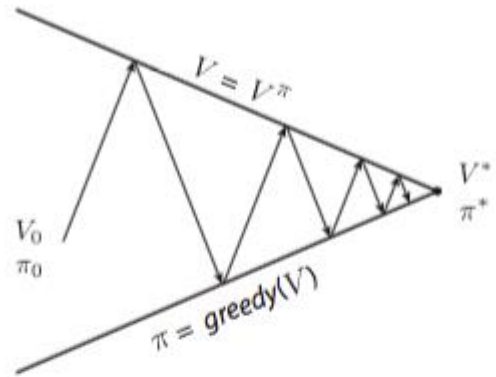
0	0.656	0.729	0.81
0.626	0.729	0	0.9
0.729	0.81	0.9	1
0.81	0.9	1	0

0.590	0.656	0.729	0.81
0.626	0.729	0	0.9
0.729	0.81	0.9	1
0.81	0.9	1	0

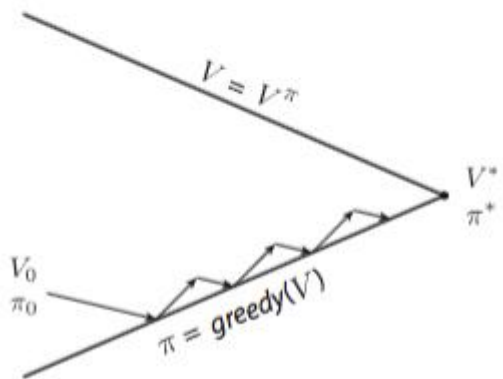
Policy

↓	→	→	↓
↓	↓		↓
↓	↓	↓	↓
→	→	→	

Value Iteration



(b)



(d)

• ביצוע לסירוגין של הערכה ושיפור, אך כל פעם צעד אחד (אין צורך לבצע שיפור מדיניות מלאה).

• למעשה, איננו צריכים בכלל מדיניות די לנו בטבלת ערכים, והמדיניות היא לבחור את הפעולה שתביא אותנו למצב עם הערך הכי גבוה.

• אנחנו מעדכנים את הערך של מצב מסויים על פי הערך של המצבים הבאים, למרות שהערך הזה הוא ערך לא מדוייק (ובתחילה הוא בכלל רנדומלי).

• יחד עם זאת לאחר מספיק איטרציות הערכים מתכנסים לערך הנכון.

Value Iteration - Bootstrapping

- לפעולה של עדכון הערכים על סמך ערכים משוערים קוראים Bootstrapping. משיכת עצמנו למעלה באמצעות שרוכי הנעליים.
- The term “bootstrapping” originated with a phrase in use in the 18th and 19th century: “to pull oneself up by one’s bootstraps.” Back then, it referred to an impossible task. Today it refers more to the challenge of making something out of nothing.



Value Iteration

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation

Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\text{terminal}) = 0$

Loop:

```

|  $\Delta \leftarrow 0$ 
| Loop for each  $s \in \mathcal{S}$ :
|    $v \leftarrow V(s)$ 
|    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$ 
|    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$ 
until  $\Delta < \theta$ 

```

Output a deterministic policy, $\pi \approx \pi_*$, such that

$$\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$$

```

1 # Value iteration for Deterministic environment
2
3 # Initialization
4 for all s in S:
5     V(s) = random(float) except V(terminal) = 0
6
7 def Value_Iteration(V):
8     accuracy = 0.0001 (small number)
9     acc = 1
10
11     while acc > accuracy:
12         acc = 0
13         for s in all states:
14             best_value = min value
15             for action in legal_actions(s):
16                 s', r = environment(s, action)
17                 new_value = r + gamma * V(s')
18                 best_value = max (best_value, new_value)
19             old_value = V(s)
20             V(s) = best_value
21             acc = max(acc, abs(old_value - best_value))
22
23     return Policy_improvement(V)
24

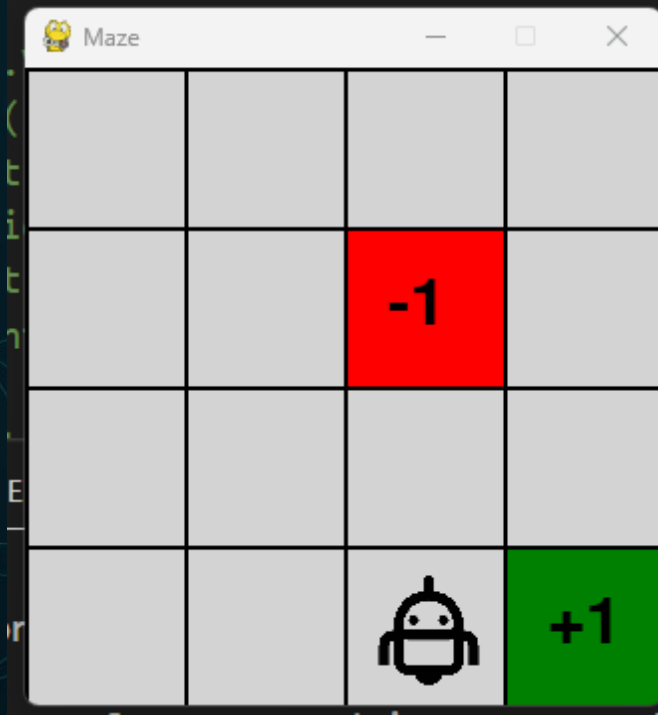
```

תרגיל GridWorld

• בממשק GridWorld עליכם להוסיף את הפונקציה:

```
def Value_Iteration()
```

• ולהריץ אותה.



הדגמה של Value Iteration

- הדגמה באמצעות פאזל מהספרים – 8 מספרים.
- מספר המצבים הוא $9! = 362880$.
- נשמור את טבלת הערכים באמצעות מילון שבו המפתח הוא המצב state והערכים הם value.
- נשמור את כל המצבים, כולל מצבים שאינם אפשריים במשחק (לכן בפועל יש פחות מצבים).
- נבצע Value Iteration על כל המצבים במילון.
- בסופה של ההרצה נקבל פתרון לכל המצבים האפשריים במשחק $3 * 3$.
- סה"כ עברנו על כל המצבים 17 פעמים (סה"כ $362,880 * 17 = 6,168,960$ איטרציות).

Model free

Value Iteration מחייב אותנו לדעת מהו המודל של הסביבה. כלומר, מה יהיה המצב הבא בעקבות כל פעולה שהסוכן ינקוט בה.

אולם, במקרים רבים הסוכן אינו מכיר את המודל, למשל במשחק מול יריב או סביבה הסתברותית (משחקי קוביות).

בשיעורים הבאים נראה את האלגוריתמים ללמידת חיזוק ללא ידיעת המודל.



קריית החינוך
כארק המדע
בית לערכים
למצימות ולחדשנות

