תכנון וקבלת החלטות אוטומטיים

תרגיל מספר 2, אלגוריתמים לפתרון MDP

כללי – בתרגיל זה עליכם לממש אלגוריתמים לפתרון MDP. כל האלגוריתמים יתבססו על חישוב פונקציית ערך (value function) וחישוב מדיניות חמדנית עבור פונקציית הערך שחישבתם. במהלך התרגיל תשתמשו בשלד פרויקט מוכן המכיל הגדרות עבור מודל MDP וכולל מספר שיטות חסרות אותם תצטרכו להשלים. כמו כן, בתרגיל זה ניתנה לכם גם המחלקה RaceViewer המאפשרת סימולציה ויזואלית של חישוב פונקציית הערך והדמייה של הרצת המדיניות.

הגשה: עליכם להגיש את התרגיל עד ה-7 ביוני, בשעה 23:59.

1. מימוש אלגוריתם Value Iteration במחלקה ValueFunction
   1. ממשו במחלקה ValueFunction את השיטה ValueIteration. השיטה מריצה את האלגוריתם הבא:

foreach s

V(s) = 0

repeat

foreach s

Δs = update(s) \\\*

until maxsΔs < ϵ

update(s)

foreach a

Q(s,a)=R(s,a)+γ∑s' tr(s,a,s')V(s')

Δs = | argmaxaQ(s,a) – V(s) |

V(s)=argmaxaQ(s,a)

return Δs

* 1. על השיטה להחזיר (במשתני ה-out) את מספר עדכוני הפונקציה שביצעה (השורה המסומנת ב-\*), כמדד לכמות החישובים, וכמו כן את זמן הריצה (wallclock time). עליכם לממש זאת בכל שיטות חישוב פונקציית הערך, על מנת שנוכל להשוות בין הסיבוכיות של השיטות השונות.
  2. ממשו את השיטה ValueAt, המחזירה את הערך הנוכחי של מצב s
  3. ממשו את השיטה GetAction המחזירה את הפעולה המיטבית עבור מצב s על פי פונקציית הערך.

1. מימוש השיטה ComputeAverageDiscountedReward במחלקה Domain
   1. על מנת לחשב את איכות הפונקציה, נריץ מספר ניסויים בהם נעקוב אחר המדיניות ממצב ההתחלה עד למצב המטרה, ונספור את סכום ה-discounted rewards.
   2. כל ניסוי יתנהל על פי השיטה הבאה

s = s0

r = 0

i = 0

while(s is not in SG)

a=π(s)

r += γiR(s,a)

i++

choose s' from tr(s,a,\*)

s = s'

* 1. הריצו cTrials ניסויים וחשבו את ממוצע הפרסים עבור כל הניסויים ע"פ הנוסחה ADR =1/n ∑j=1..n∑i=1..γi ri
  2. שימו לב – יתכן כי המדיניות לא תביא אותנו אל מצב המטרה, לכן יש להפסיק את הניסוי אם עברו cStepsPerTrial צעדים.

1. מימוש השיטה PrioritizedValueIteration במחלקה ValueFunction
   1. כעת נחשב את פונקציית הערך בצורה מעט יותר יעילה. נחזיק תור עדיפויות של המצבים, כאשר העדיפות של כל מצב תאותחל לפרס המיידי המקסימלי של אותו מצב.
   2. נבצע את האלגוריתם הבא:

while(max priority in PQ is more than ϵ)

s = PQ.ExtractMax

Δs=update(s)

foreach s' ∈ preds(s)

priority(s') = max( priority(s'), γΔs)

* 1. לנוחיותכם, המחלקה Heap מאפשרת את כל הפונקציונליות הנדרשת, כולל שינוי ה-priority של מצב.

1. מימוש השיטה RealTimeDynamicProgramming במחלקה ValueFunction
   1. כעת נריץ trials ונעדכן את פונקציית הערך בהתאם ל-trial אותו ביצענו.
   2. שימו לב – אין לאתחל ערך עבור כל מצב, אלא להשתמש בערך ברירת מחדל אופטימי (כלומר upper bound על הערך האופטימלי) עבור מצבים שטרם ראינו. ניתן, עבור העולם עליו אנו רצים, להשתמש בפרס המיידי המקסימלי הגלובלי כערך האופטימי עבור כל המצבים – Rmax=maxs,aR(s,a).
   3. נריץ את האלגוריתם הבא:

s=s0

stack = the empty stack

while(s is not in SG)

update(s)

stack.push(s)

a\*=maxaQ(s,a)

choose s' from tr(s,a,\*)

s=s'

while(!stack.empty)

s=stack.pop

update(s)

שימו לב – אין לשלוח קטעי קוד מתלמיד לתלמיד. אל תעתיקו!

שאלות ובעיות יש להפנות [ל-shanigu@bgu.ac.il](mailto:ל-shanigu@bgu.ac.il)

בהצלחה!