מטלת מנחה (ממ"ן) 13 - להרצה

הקורס: **20551 – מבוא לבינה מלאכותית**

חומר הלימוד למטלה: פרקים 1-5

מספר השאלות: 4 **משקל המטלה:** 4

סמסטר: 2021א **מועד אחרון להגשה:** 28.11.2020

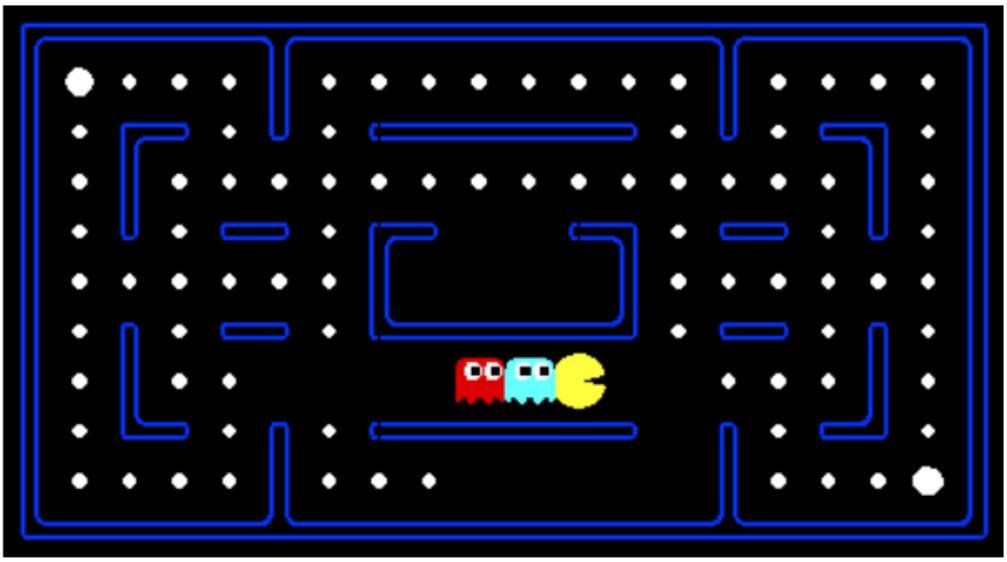
(אב)

קיימות שתי חלופות להגשת מטלות:

- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס
- שליחת מטלות באמצעות הדואר או הגשה ישירה למנחה במפגשי ההנחיה

הסבר מפורט ב"נוהל הגשת מטלות מנחה"

Pac-Man מרובה סוכנים - Multi-Agent Pac-Man



Pacman, now with ghosts.

הקדמה

במטלה זו, עליכם לתכנן סוכנים עבור הגירסה הקלאסית של Pacman עם רוחות רפאים. לצורך כך תממשו את אלגוריתם מינימקס, אלפא-ביתא ותתנסו בתכנון פונקציית הערכה.
כקודמתה, גם מטלה זו פותחה באוניברסיטת ברקלי על-ידי John DeNero ו-Dan Klein.

הקוד עבור מטלה זו לא השתנה בהרבה ביחס לזה של המטלה הקודמת, אך רצוי מאד כי תתקינו אותו מחדש. בכל אופן, אתם יכולים להשתמש בקבצים search.py ו-searchAgents.py שלכם בכל דרך שתרצו.
את כל הקוד והקבצים הנלווים ניתן להוריד כקובץ zip.

הקוד עבור המטלה מכיל את הקבצים הבאים:

קבצים שיש לקרוא:

multiAgents.py	כל סוכני החיפוש (multi-agent) שלכם יימצאו שם.
pacman.py	הקובץ העיקרי שמריץ משחקי פקמן. הקובץ מתאר את הטיפוס GameState שתשתמשו בו הרבה במטלה זו
game.py	הלוגיקה שמאחורי אופן פעולת עולם הפקמן. קובץ זה מתאר מספר טיפוסי עזר כגון AgentState, Agent, Direction, Grid
util.py	מבני נתונים שימושיים למימוש אלגוריתמי חיפוש

קבצים שאין צורך להתעמק בקוד שלהם:

graphicsDisplay.py	גרפיקה עבור פקמן
graphicsUtils.py	תמיכה לגרפיקה של פקמן
textDisplay.py	גרפיקת אסקי לפקמן
ghostAgents.py	סוכנים לבקרה על רוחות רפאים
keyboardAgents.py	ממשקי מקלדת לבקרה על פקמן
layout.py	קוד לקריאת קבצי פריסה (layout) ואחסון תוכנם

עליכם להגיש

במהלך ביצוע המטלה יש למלא חלקים חסרים בקוד של multiAgents.py.
עליכם לשלוח למנחה קובץ זה (בלבד) וקובץ עם שמכם, ת.ז. והתיעוד.
הנכם מתבקשים לא לשנות שמות של פונקציות או מחלקות הנתונות בקוד.

Pac-Man מרובה סוכנים

תחילה הריצו משחק של Pac-Man הקלאסי:

```
python   pacman.py
```

כעת הריצו את ReflexAgent הנתון ב-multiAgents.py :

```
python  pacman.py  -p  ReflexAgent
```

שימו לב שהוא משחק די גרוע אפילו עבור פריסות (layouts) פשוטות :

```
python  pacman.py  -p  ReflexAgent  -l  testClassic
```

בדקו את הקוד (ב-multiAgents.py) וודאו שאתם מבינים את אופן פעולתו.

שאלה 1 (23%)

שפרו את ReflexAgent ב-multiAgents.py כדי שישחק ברמה סבירה.

(בקוד הנתון עבור סוכן של תגובה-פשוטה (reflex agent) יש כמה דוגמאות שיכולות להיות לעזר של שיטות שמתשאלות את ה-GameState).

סוכן טוב של תגובה פשוטה צריך לקחת בחשבון הן מיקומים של אוכל והן מיקומים של רוחות רפאים. הסוכן שלכם אמור לאפס בקלות את פריסת ה-testClassic :

```
python  pacman.py  -p   ReflexAgent  -l   testClassic
```

נסו את הסוכן תגובה-פשוטה שלכם על פריסת ברירת המחדל -mediumClassic עם רוח רפאים אחת או שתיים (וכבו את האנימציה כדי להאיץ את הריצה) :

```
python  pacman.py   --frameTime   0   -p   ReflexAgent   -k   1

python  pacman.py   --frameTime   0   -p   ReflexAgent   -k   2
```

קרוב לוודאי שהסוכן שלכם ימות לעתים קרובות כאשר יש שתי רוחות רפאים על לוח ברירת המחדל, אלא אם כן פונקציית ההערכה שלכם טובה למדי.

שימו לב :

- לא יתכן שיהיו לכם יותר רוחות רפאים מאשר מאפשרת ה-layout.
- פונקציית ההערכה שתכתבו מעריכה זוגות של מצב-פעולה; בהמשך המטלה תעריכו מצבים.

אופציות:

הרוחות שהן ברירת המחדל מתנהגות באופן אקראי. אתם יכולים בשביל הכיף לשחק עם רוחות קצת יותר חכמות בעזרת -g DirectionalGhost . אם האקראיות מונעת מכם להסיק האם הסוכן שלכם השתפר, אתם יכולים להשתמש ב -f כדי להריץ עם גרעין אקראי קבוע (אותן בחירות אקראיות בכל משחק). אתם יכולים לשחק משחקים מרובים ברצף עם -n. כבו את הגרפיקה עם q– כדי להריץ הרבה משחקים במהירות.

ניקוד מלא ינתן לסוכן שיכול לאפס במהירות את פריסת ה- openClassic עשר פעמים, מבלי למות יותר מפעמיים או לנוע הלוך ושוב באופן חוזר ונשנה בין שני מיקומים מבלי להתקדם.

```
python pacman.py -p ReflexAgent -l openClassic -n 10 –q
```

שאלה 2 (32%)

כעת עליכם לכתוב קוד עבור סוכן חיפוש בתנאי יריבות במחלקה MinimaxAgent אשר ב-multiAgents.py. סוכן המינימקס שלכם אמור לעבוד לכל מספר של רוחות רפאים, לכן עליכם לכתוב אלגוריתם שיהיה קצת יותר כללי מזה שבספר הלימוד. ובמיוחד, לעץ המינימקס שלכם יהיו מספר רמות Min (אחת לכל רוח רפאים) עבור כל רמה של שחקן Max.

בנוסף, הקוד שתכתבו צריך לפתח את עץ המשחק לעומק כלשהו.

ערכי העלים בעץ המינימקס שלכם ייקבעו על ידי self.evaluationFunction הנתונה, כאשר ברירת המחדל שלה היא scoreEvaluationFunction. המחלקה MinimaxAgent יורשת את MultiAgentAgent , שנותנת גישה ל- self.depth ו-self.evaluationFunction . וודאו כי קוד המינימקס שלכם משתמש במשתנים אלה כשצריך, שכן משתנים אלה מקבלים ערך כתוצאה מהאפשרויות של שורת הפקודה (command line).

חשוב:

שלב אחד בחיפוש נחשב כמהלך אחד של Pac-Man וכל התגובות של רוחות הרפאים, לכן עומק 2 בחיפוש פירושו שני מהלכים של Pac-Man ושל כל רוח רפאים.

רמזים והערות

- פונקציית ההערכה עבור שאלה זו כתובה כבר (self.evaluationFunction). אינכם אמורים לשנות אותה, אך שימו לב לכך שכעת אנו מעריכים מצבים ולא פעולות, כפי שעשינו עבור הסוכן של תגובה פשוטה. סוכנים המסתכלים-קדימה מעריכים מצבים עתידיים בעוד שסוכנים של תגובה פשוטה מעריכים פעולות מהמצב הנוכחי.
- ערכי המינימקס עבור המצב ההתחלתי בסידור ה-minimaxClassic הם 9,8,7,-492 לעומקים 1,2,3,4 בהתאמה. שימו לב לכך שסוכן המינימקס שלכם ינצח לעתים קרובות (665/1000 משחקים) למרות שהחיזוי שלו הוא שמצבים בעומק 4 הם גרועים.

```
python pacman.py -p MinimaxAgent -l minimaxClassic -a depth=4
```

- כדי להגדיל את עומק החיפוש שתוכלו להשיג באמצעות הסוכן שלכם, הסירו את הפעולה Directions.STOP מהרשימה של Pac-Man של כל הפעולות האפשריות עבורו. עומק 2 יהיה מהיר יחסית אך עומק 3 או 4 יהיה איטי. בשאלה הבאה נטפל במהירות החיפוש.
- פקמן הוא תמיד סוכן 0, והסוכנים נעים בסדר עולה של האינדקסים שלהם.
- כל המצבים במינימקס צריכים להיות GameStates המועברים ל-getAction או נוצרים באמצעות GameState.generateSuccessor.
- עבור לוחות גדולים יותר כגון openClassic ו-mediumClassic (ברירת המחדל), תמצאו לנכון ש-Pac-Man עושה חיל ב"לא למות" אך גרוע ב"לנצח". לעתים קרובות הוא יסתובב סביב עצמו מבלי להתקדם. הוא אפילו עלול להסתובב ממש ליד נקודה מבלי לאכול אותה, משום שאינו יודע לאן ללכת לאחר שיאכל אותה.
- כאשר Pac-Man מאמין שמותו בלתי נמנע, הוא ינסה לסיים את המשחק מהר ככל האפשר, בגלל העונש הקבוע עבור הישארות בחיים. לעיתים זוהי הבחירה השגויה לטיפול ברוחות רפאים אקראיות אך סוכני מינימקס מניחים תמיד את הגרוע ביותר:

```
python pacman.py -p MinimaxAgent -l trappedClassic -a depth=3
```

וודאו כי אתם מבינים מדוע Pac-Man נחפז לרוח הרפאים הקרובה ביותר במקרה זה.

שאלה 3 (23%)

צרו ב-AlphaBetaAgent סוכן חדש המשתמש בגיזום אלפא-ביתא כדי לסרוק את עץ המינימקס בצורה יעילה. שוב, האלגוריתם שלכם יהיה מעט יותר כלי מהפסאודו קוד שבספר הלימוד, כך שחלק מהאתגר הוא להרחיב את הרעיון שבגיזום אלפא-ביתא כך שיתאים לסוכני מינימום רבים.

עליכם לראות שיפור במהירות (למשל, אולי אלפא-ביתא בעומק 3 ירוץ באותו זמן של מינימקס בעומק 2). המצב האידיאלי הוא שעומק 3 ב-smallClassic ירוץ במספר שניות לכל מהלך או אף מהר יותר.

```
python pacman.py -p AlphaBetaAgent -a depth=3 -l smallClassic
```

ערכי המינימקס של AlphaBetaAgent צריכים להיות זהים לערכי המינימקס של MinimaxAgent, למרות שהפעולות שהוא בוחר יכולות להשתנות בגלל התנהלות שונה במקרים של החלטות שונות לשבירת שוויון. שוב, ערכי המינימקס של המצב ההתחלתי בסידור ה-minimaxClassic הם 492-,7,8,9 עבור העומקים 1,2,3,4 בהתאמה.

שאלה 4 (22%)

האלגוריתמים מינימקס ואלפא ביתא מניחים שהשחקן משחק נגד יריב המקבל החלטות אופטימליות. אך זה אינו תמיד המצב, למשל במשחק איקס עיגול.

בשאלה זו, עליכם לממש את ה- ExpectimaxAgent המשמש למידול התנהגות אקראית של סוכנים שבחירותיהם יכולות להיות לא אופטימליות.

מומלץ לבדוק את הקוד שלכם על עצי משחק קטנים ולאחר שהאלגוריתם ירוץ נכון עבורם, תוכלו לבחון את הצלחתו על פקמן.

כאשר אתם מחשבים ערכים ממוצעים, הקפידו להשתמש במספרים עשרוניים ולא במספרים שלמים. חילוק בשלמים בפייתון יגרום למשל לכך ש-1/2=0 ואילו במספרים עשרוניים 1.0/2.0=0.5.

רוחות אקראיים הם כמובן סוכני מינימקס לא אופטימליים, ולכן מידול שלהם בעזרת חיפוש מינימקס, לא בהכרח יתאים.

ExpectimaxAgent לא תיקח את המינימום על כל הפעולות של הרוחות, אלא את התוחלת לפי המודל של הסוכן שלכם לאיך הרוחות מתנהגות.

כדי לפשט את הקוד, הניחו שהקוד ירוץ רק נגד יריב שבוחר **אקראית** מבין הפעולות המוחזרות ע"י getLegalAction.

כדי לבדוק איך ה-ExpectimaxAgent מתנהג בפקמן הריצו:

```
python pacman.py -p ExpectimaxAgent -l minimaxClassic -a depth=3
```

עכשיו תוכלו לראות התנהגות אדישה יותר בקרבת הרוחות. ביחוד אם פקמן חש שהוא עלול להילכד אבל יכול להימלט כדי לחטוף עוד כמה חתיכות מזון, או לפחות ינסה.

חקרו את התוצאות של שני התסריטים הבאים:

```
python pacman.py -p AlphaBetaAgent -l trappedClassic -a depth=3 -q -n 10
python pacman.py -p ExpectimaxAgent -l trappedClassic -a depth=3 -q -n 10
```

אתם אמורים לראות ש- ExpectimaxAgent שלכם מנצח לפחות חצי מהזמן, בעוד AlphaBetaAgent תמיד מפסיד. הסבירו מדוע ההתנהגות כאן שונה מאשר במינימקס.

מימוש נכון של Expectimax יוביל לכך שפקמן יפסיד לעתים. זו לא בעיה: זו התנהגות נכונה שתעבור את המבדקים שלנו.