מבוא לבינה מלאכותית - 236501

תרגיל בית 2

מגישים:

יקיר חלץ 305028441

גל פלייסיג 302912985

חלק א׳ – היכרות עם הקוד והמשחק

1. בוצע.
2. בוצע.
3. השחקן הבסיסי ReflexAgent עובד בצורה הבאה: הוא בוחן את כל המהלכים החוקיים שהוא יכול לבצע מהמצב הנוכחי, ולכל אחד מהם הוא מעריך את טיב המהלך על-ידי מתן ציון בצורת מספר לכל מהלך חוקי (פירוט כיצד – בהמשך). אחר-כך הוא שומר את הערך המקסימלי מבין הציונים. בשלב הבא הוא מחפש את כל המהלכים שנותנים ערך מקסימלי זה (יתכן יותר ממהלך אחד), ובוחר בצורה רנדומלית מבניהם את המהלך הבא לביצוע.

היוריסטיקה שבה הוא משתמש נעזרת בנתון שידוע לנו אשר מציג את תוצאת השחקן לאחר כל מהלך (score): המהלך בעל ערך היוריסטיקה הטוב ביותר הוא המהלך אשר בהינתן ששוחק ישיג את התוצאה הכוללת (score) הגבוהה ביותר עבור השחקן, מבין כל יתר אפשרויות המשחק מאותה נקודה.

חלק ב׳ – בניית סוכן משופר

1. היוריסטיקה תוגדר באופן הבא:

ראשית, בדוק את כל המהלכים החוקיים מהמהלך הנוכחי, ולכל מהלך חוקי כזה שמור את שהמצב שמגיעים אליו לאחר ביצוע המהלך. הוסף נקודות לכל אחד מהמצבים - לפי הסעיפים הבאים (בצורה בלתי תלויה ביתר המצבים):

* 1. בדוק את המרחק (מנהטן) של המיקום של פקמן במצב הנוכחי מהרוח הכי קרובה אליו. אם המרחק קטן ממש מ-fear\_factor אז הוסף למצב ניקוד השווה למרחק שחושב פחות ghost\_proximity\_penalty (שמוגדר כרגע להיות 1000). אחרת אל תוסיף נקודות כלל והמשך להוסיף נקודות לפי הסעיפים הבאים.
  2. הוסף את הניקוד של המצב הנוכחי.
  3. בדוק לכל משבצת את המרחק שלה (בפועל בצעדים ולא מנהטן) לחתיכת האוכל הקרובה ביותר אליה, והוסף למשבצת ניקוד השווה למרחק זה בסימן מינוס.
  4. (בהמשך אפשר להוסיף שאם מצב הרוח הוא שהיא לבנה אז תלך אליה – במקום השלב הראשון, אחרת תתרחק).

לסיום לאחר שסכמנו את כל הנקודות שהמצב קיבל מכל אחד מהסעיפים, מתקבלת תוצאה וזה מהווה הניקוד של המצב.

השתמשנו באלמנטים של מרחק מרוח, ניקוד במשבצת הנוכחית (שמגלם בתוכו אוכל/קפסולה/אכילת רוח במצב הנוכחי), ומרחק מחתיכת האוכל הקרובה ביותר – הנמדד בצעדים.

הערות:

\* המרחק בסעיף a מחושב באמצעות מרחק מנהטן, ואילו המרחק בסעיף c מחושב באמצעות מרחק בפועל בצעדים.

\* בכל אחד מהסעיפים שבו צריך לעשות בחירה – אם יש כמה מצבים עם ניקוד זהה – בוחרים רנדומלית מבין המצבים בעלי הניקוד הגבוה ביותר, כמו שנעשה באלגוריתם המקורי

1. המוטיבציה להגדרה מהסעיף הקודם היא שאנחנו רוצים לבנות פונ׳ יוריסטית שמתחשבת בגורמים נוספים מלבד הגורם של הניקוד במצב הבא, ובייחוד בשני גורמים ספציפיים – האחד הוא מרחק מרוחות. פקמן יפסל אם נפגש ברוח, ולכן בעדיפות הגבוהה ביותר אנו רוצים להימנע ממצב שבו רוח תאכל את פקמן. השני הוא מרחק מחתיכת האוכל הקרובה ביותר גם אם היא לא נמצאת בסביבתו הקרוב של פקמן. זאת מכיוון שאנו לא רוצים לאפשר מצב של תקיעה כאשר אין בסביבתו הקרוב של פקמן אוכל, אלא שהוא יבצע תנועה לכיוון חתיכת האוכל הקרובה ביותר בכל רגע נתון ובכך יעלה את הניקוד שלו ויסיים את המשחק במהרה, ולא יהיה נתון לסיכון שבפגישה עם רוח כאשר הוא עושה צעדי הלוך ושוב במקום.

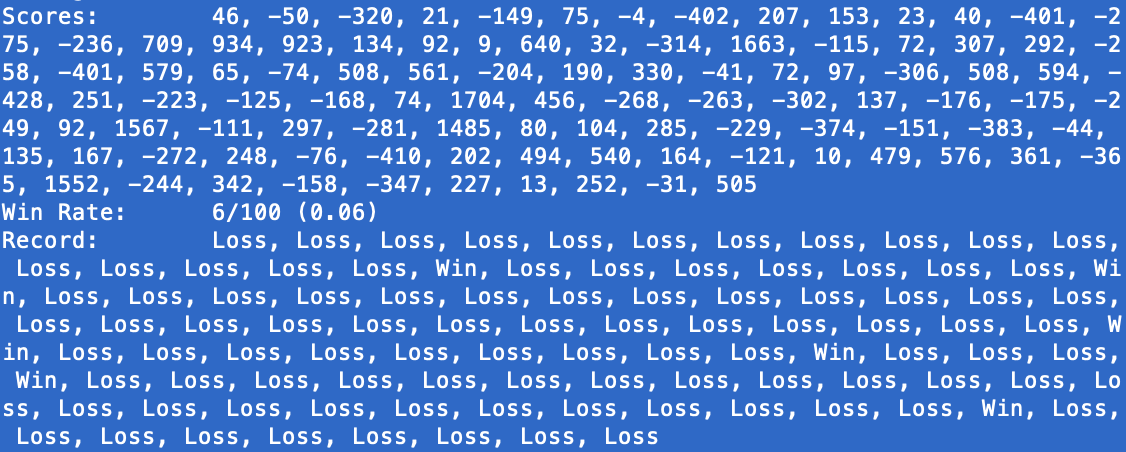
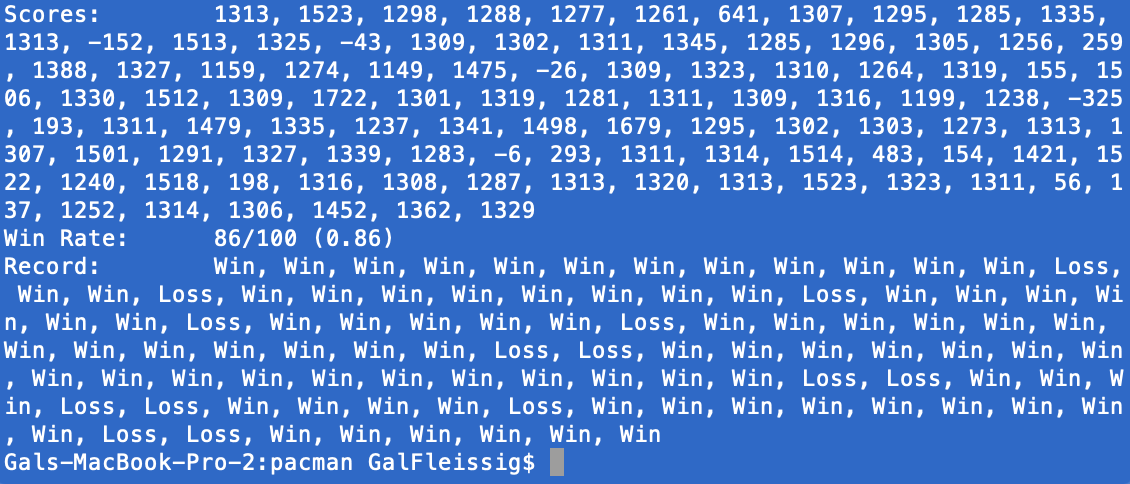
הסבר לכל פרמטר המופיע בחישוב על פי הסעיף המתאים:

* 1. לעומת זאת האלגוריתם שלנו בהינתן שיש רוח קרובה (מחושב עם מרחקי מנהטן) – יעדיף בעדיפות ראשונה להתרחק ממנה עד שהוא רחוק ממנה מספיק (כמובן כתלות ב-fear\_factor). כאן נציין כי ה-ghost\_proximity\_penalty נקבע להיות 1000 על-מנת ליצור הבחנה ברורה בין מצב שקרוב לרוח לבין מצב שאינו קרוב לרוח, כאשר אם השחקן קרוב מדי לרוח העדיפות הראשונה שלו תהיה לברוח ממנה.
  2. בנוסף הוא יעדיף משבצות עם score כולל גבוה יותר (בדומה ליוריסטיקה הפשוטה), וזאת מכיוון שאנו רוצים לתת עדיפות לאוכל, שכן משבצת סמוכה עם אוכל תועדף על פני משבצת סמוכה ללא אוכל שכן המשבצת עם האוכל תניב ניקוד גבוה יותר למצב. עוד נגדיש כאן כי אם נוצר מצב שבו פקמן נמצא במרחק משבצת אחת מרוח במצב אכיל, הוא יעדיף לאכול אותה בעקבות סעיף b שכן המצב שאחרי האכילה יניב לו ניקוד גבוה.
  3. וכמו-כן אם אין בסביבתו המיידית אוכל כלל יעדיף פקמן ללכת לכיוון המשבצת עם חתיכת האוכל הקרובה ביותר (וזאת בניגוד ליוריסטיקה הפשוטה כפי שצוין לעיל) וזאת בהתבסס על מרחקים שנמדדים בצעדים ממש ולא בהתבסס על מרחקי מנהטן.

אנו צופים שהיא תשפר את ביצועי השחקן ביחס ליוריסטיקה scoreEvaluationFunction בה השתמש השחקן הפשוט עד כה מכיוון שהשחקן הפשוט העדיף מהלכים שיובילו אותו למשבצות עם אוכל או קפסולה, ואם כל המשבצות מסביב המשבצת הנוכחית שבה הוא נמצא ריקות מאוכל או קפסולה הוא ילך בצורה רנדומלית לאחת מהמשבצות השכנות, כלומר עשוי לא להתקדם לקראת אוכל או קפסולה נוספים אלא אם כן הפונ׳ הרנדומלית תפנה אותו לשם בשלב כלשהו.

היוריסטיקה המשופרת שלנו לעומת זאת מתחשבת באלמנטים נוספים כגון מרחק מרוחות, מרחק מחתיכת האוכל הקרובה ביותר **גם אם היא לא נמצאת בסביבה המיידית של פקמן** כפי שצוין, ועוד, ובכך מנצלת את המידע שברשותה בכדי לשחק בצורה חכמה יותר.

ניתן לראות כי אם בממוצע ה-ReflexAgent עם היוריסטיקה הפשוטה מנצח כ-6 מתוך 100 משחקים, היוריסטיקה המשופרת שלנו מנצחת כ-86 מתוך 100 משחקים (מבוסס על הרצה ללא גרפיקה של 100 משחקים עם כל אחת מהיוריסטיקות).



1. מימוש בקוד.

חלק ג׳ – בניית סוכן Min-Max

1. כאשר ביצירת העץ אנו יוצרים מספר שכבות min (אחת לכל רוח רפאים) עבור כל שכבת max (המייצגת את פקמן), אנו למעשה בכך קובעים סדר מסוים לריצת הרוחות: נניח לשם הפשטות שישנן שתי רוחות במשחק, אחת סגולה ואחת כתומה. אם למשל מיד לאחר שכבת ה-max תגיע שכבת ה-min של הרוח הסגולה, ולאחריה שכבת ה-min של הרוח הכתומה, ואז שכבת ה-max הבאה של פקמן, אזי אנו מניחים כי הרוח הסגולה ״משחקת לפני״ הרוח הכתומה. ואז הרוח הכתומה מקבלת החלטות לגבי תנועתה הבאה בהתאם לידע שיש לה על התנועה שזה עתה ביצעה הרוח הסגולה. אבל הנחה זו לא בהכרח נכונה, שכן בפועל אין בהכרח רוח מסוימת ש״תשחק לפני״ רוח אחרת.
2. מימוש בקוד.
3. במקום ליצור שכבה נוספת לכל רוח, ניתן ליצור שכבר אחת לכלל הרוחות. שכבה זו תהיה כמובן שכבת min, והצעדים בה יהיו כל הקומבינציות האפשריות של צעד אחד של כל רוח. לכן אם יש n רוחות, ונניח שלכל רוח יש m צעדים אפשריים, אזי בשכבה יחידה זו יהיו m^n צעדים, שהן בחירה של אחד מתוך m צעדים לרוח הראשונה, כפול בחירה של אחד מתוך m צעדים לרוח השנייה, וכן הלאה - ואנו נבחר מתוך כלל צעדים אלה את הערך המינימלי של מינימקס.

היתרון של שיטה זו הוא שיש שכבה אחת ויחידה לכלל הרוחות, וזה מסיר את ההנחה - שהסברנו שהיא לא בהכרח נכונה – על כך שרוח מסוימת משחקת לפני רוח אחרת ויכולה לקבל החלטות בהתבסס על צעד הרוח ששיחקה לפניה. החסרון של שיטה זו הוא

באופן דואלי, החסרון של השיטה הקודמת מסעיף 1 הוא ההנחה שהסברנו שהיא לא בהכרח נכונה – על כך שרוח מסוימת משחקת לפני רוח אחרת ויכולה לקבל החלטות בהתבסס על צעד הרוח ששיחקה לפניה. היתרון הוא...

חלק ד׳ – בניית סוכן Alpha – beta

1. מבנה העץ החדש שהגדרנו אינו משפיע על אלגוריתם אלפא-בטא, שכן ההבדל בינו לבין עץ סטנדרטי כפי שנלמד הוא שיש בו כמה שכבות של ״יריבים״ בהם היריב מתעדף ערך מינימלי – והן באות בזו אחר זו – והבדל זה אינו משפיע על הגיזום, שכן בדומה למבנה העץ הסטנדרטי, גם במבנה העץ החדש בבואנו לבצע גיזום אלפא-בטא אנחנו מסתכלים על המהלכים הבאים לביצוע מהשורש, ובהינתן שיש לנו תוצאה מסוימת, אנחנו יכולים לתת תנאים על טיב תת העץ הבא לאחר שגילינו רק חצי ממנו, ובמקרה שבו התוצאה העתידית הכי טובה שלו תהיה בוודאות פחות טובה מהתוצאה שכבר חשפנו – אף לבצע גיזום של הענף.

למשל, אם אנחנו מחפשים את הערך המקסימלי וקיבלנו שבתת עץ אחד הערך הוא +2 ובתת-העץ הבא הערך יהיה קטן או שווה ל-0 (על פי הענף השמאלי בלבד של תת העץ), נוכל לגזום את תת העץ הבא כולו.

1. מימוש בקוד.
2. (א) מבחינת זמן ריצה, מההרצאות ראינו כי מינימקס מפתח b^d צמתים. לעומת זאת, מספר הצמתים שאלפא בטא יפתח תלוי בסידור הצמתים: בסידור אופטימלי יפתח אלפא בטא b^0.5d צמתים. בסידור אקראי יפתח אלפא בטא b^0.75d צמתים. ואילו במקרה הגרוע שבו הצמתים מסודרים מהקטן לגדול (כאשר מחפשים מקסימום) או מהגדול לקטן (כאשר מחפשים מינימום) – יפתח אלפא בטא b^d צמתים - וסיבוכיות הזמן היא כשל minimax.

(ב) מבחינת בחירת מהלכים, שני הסוכנים יתנהגו בצורה זהה, וזאת מכיוון שבה״כ אם הסוכן מעוניין בערכים מקסימליים, אזי הוא תמיד יבחר במהלך בעל הערך המקסימלי – גם במינימקס וגם באלפא-בטא: ההבדל הוא שבמינימקס הוא יפתח את כל תתי העצים מהצומת הנוכחי ורק אז יבצע את המהלך של תת העץ שהחזיר ערך מקסימלי, ואילו באלפא-בטא הוא יפתח את תת העץ הראשון ובהמשך כל תת עץ שיניב בוודאות תוצאה גרועה יותר – הוא לא יפתח את כולו – ויבצע גיזום בשלב מוקדם. המהלך של תת עץ זה לא היה נבחר גם במינימקס, ולא יבחר גם כאן באלפא-בטא. אם התוצאה הטובה ביותר נמצאת בילד האחרון של הצומת הנוכחי, אלפא-בטא יפתח אותו **במלואו** ויבחר בו בסופו של דבר – בדיוק כפי שמינימקס יבחר בו.