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.

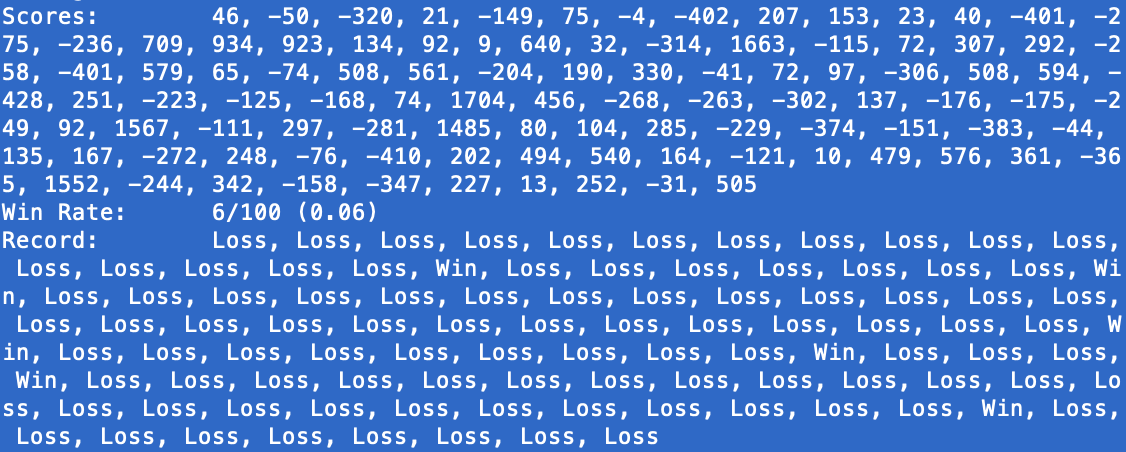
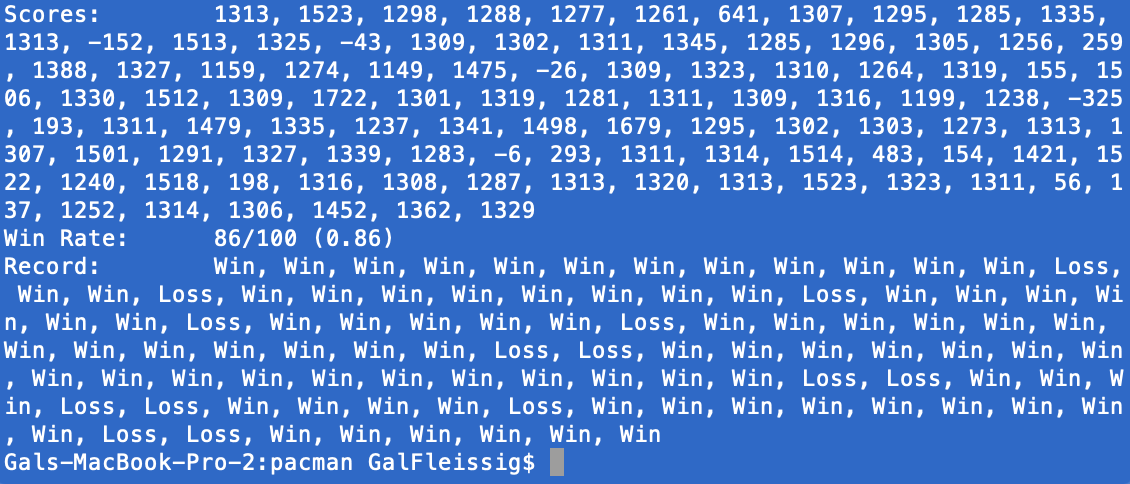
הסבר לכל פרמטר המופיע בחישוב על פי הסעיף המתאים:

* 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 בסיבוב התורות הנוכחי. 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 (שהוא גם החסרון של השיטה מסעיף 1) 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 (שהוא גם היתרון של השיטה מסעיף 1) הוא שכעת החישוב של ערך המינימקס של שכבת הרוחות עשוי לקחת זמן רב שכן יש לחשב את ערך המינימקס של m^nצמתים בגרף. לעומת זאת, בשיטה מסעיף 1 החישוב (עבור רוח אחת) יקח זמן מועט יותר: בכל אחת מכמה שכבות ה-min יש לחשב את ערך המינימקס של m צמתים בלבד (עבור רוח בודדת). אמנם אם נסתכל על חישוב ערכי המינימקס של כלל הרוחות בכל שכבות ה-min לפי השיטה בסעיף 1 נבחין כי מספר הצמתים לחישוב הוא זהה לזה שבשיטה בסעיף זה, בשיטה בסעיף 1 נוכל לדעת מה הצעד שצריכה לעשות רוח מסוימת עוד בטרם חיפשנו את הצעד שצריכה לעשות הרוח הבאה, ואם אנו מוגבלים בזמן – עדיף שיהיה לנו מידע חלקי על הצעדים המיטביים של חלק מהרוחות – מאשר שלא יהיה לנו מידע כלל על אף אחת מהרוחות, מה שעלול לקרות עם השיטה מסעיף זה.

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.

חלק ה – בניית סוכן Expectimax לרוח רנדומלית

1. מימוש בקוד.
2. השינוי העיקר ביחס לשני הסוכנים הקודמים הוא בהנחה שלנו לגבי הצעד הבא של כל אחת מהרוחות - בבואנו לבחור את הצעד הבא של פקמן. בשני סוכני ה-Minimax (ה-Minimax הרגיל וה-Minimax עם גיזום אלפא-בטא), אנחנו מניחים שהיריב (במקרה זה – הרוחות) יבחר בצעד שיביא למינימום את הניקוד של פקמן, כלומר אנו מניחים שכל אחת מהרוחות תבחר בצעד עם ערך ה-Minimax הנמוך ביותר (כפי שלמדנו בנושא). לעומת זאת, עבור הסוכן בחלק זה, אנו מניחים שההסתברות של רוח לבחור בכל אחד מהצעדים האפשריים שלה היא שווה. במילים אחרות – אין כאן הנחה שהרוח מנסה להביא למינימום את הניקוד של פקמן, אלא שהצעדים שלה הם אקראיים לחלוטין.

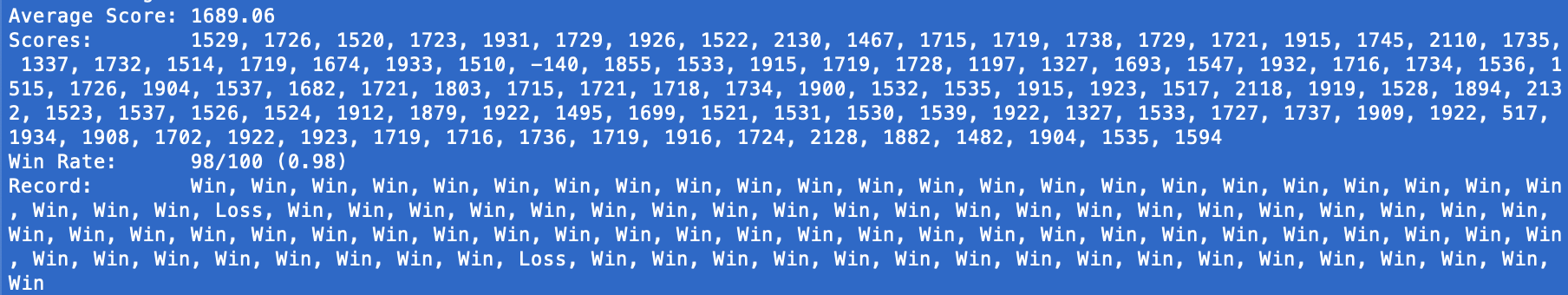
נדגיש את הנקודה: גם בסעיפים ג׳-ד׳ וגם בסעיף ה התנהגות הרוח היא זהה – בכל המקרים הללו הרוח מתנהגת לפי RandomGhost אשר מבצעת כל צעד בהסתברות שווה ביחס לשאר הצעדים. השינוי בסעיף זה הוא **בהתייחסות שלנו** למה אנו מניחים שהצעד הבא יהיה. בסוכנים הקודמים מניחים שהצעדים של היריב יהיו לרעתנו, כלומר מניחים את המקרה הגרוע עבור פקמן, ובסעיף זה אנו מניחים שההתפלגות היא שווה – כמו במציאות.

לגבי התוצאות בהרצה, אנחנו מצפים שהשחקן פקמן ינצח באחוז גבוה יותר של המקרים. זאת מכיוון שהשחקן משחק בצורה ריאלית מדי ולא בצורה פחדנית יתר על המידה אשר גורמת לו להיות זהיר מדי ועקב כך לנצח פחות פעמים. כאשר פקמן זהיר מדי הוא עלול שלא לעשות צעדים אשר ייטיבו עימו ואשר הרוח בהסתברות גבוהה תבחר אותם אלא לעשות צעדים מניעתיים ו״בורחים״ מהרוח רק בשל ההסתברות הנמוכה שהם אכן יקרו.

ניתן שתי דוגמאות של מקרים שונים שיתמכו בציפיות:

1. אם למשל לפקמן נותרה חתיכת אוכל אחרונה לאכול והוא רחוק ממנה 2 צעדים, אבל הוגדרה לו באלגוריתם מינימקס (כפי שאנו מימשנו אותו) דרגת ״פחדנות״ גבוהה יחסית, כלומר אם הוא במרחק קטן יחסית מרוח העדיפות הגבוהה ביותר שלו תהיה לברוח, אזי הוא ינסה לברוח אחורה מהרוח ואז הוא יכול להיתקל ברוח אחרת שתסגור עליו מן הצד השני. מה שיכול היה לקרות אם פקמן היה פועל לפי RandomExpectimaxAgent הוא שפקמן יכול היה ״לקחת את הסיכון״ וללכת לעבר חתיכת האוכל האחרונה, ומכיוון שההסתברות שהרוח תבחר בצעד שילך לעבר פקמן היא שווה להסתברות שהיא תבחר בצעד כלשהו אחר (ואם נניח שסך ההסתברויות שהרוח לא תלך לעבר פקמן היא גבוהה יחסית) אזי ההסתברות שפקמן היה מצליח לאכול את חתיכת האוכל האחרונה ולנצח את המשחק היא גבוהה.
2. מקרה נוסף הוא מקרה שבו פקמן קרוב מאד לקפסולה, ומן העבר השני של הקפסולה ממתינה לו רוח. אזי גם במקרה זה, אם פקמן יפחד מדי ויברח מהרוח מכיוון שהיא קרובה, אזי הוא יסוג, לא יסיים את המשחק כעת ומן הצד השני יכולה להמתין לו רוח ש״תסגור עליו״ כמו קודם ותאכל אותו. לעומת זאת, אם הוא היה מתחשב בהסתברויות שהרוח תלך לכיוונו מהצד של הקפסולה ומבין שסך ההסתברויות שהרוח לא תלך לכיוונו גבוהה מסך ההסתברויות שהרוח תלך לכיוונו - והולך לעבר הקפסולה - הוא היה מגיע אליה, ואז יכול במקום מצב שבו הוא נפסל ונאכל, לאכול את שני הרוחות, להמשיך ולסיים את המשחק בניצחון.

להלן התוצאות של 100 הרצות עם השחקן RandomExpectimaxAgent וניתן לראות שהוא קיבל את הציון 98/100, כלומר אכן הציפיה שלנו התממשה:



חלק ו׳ – בניית סוכן Expectimax לרוח לא רנדומלית

1. האסטרטגיה המלאה של הרוח DirectionalGhost היא ראשית לשמור את הפעולות החוקיות שלה מהמצב הנוכחי. כעת, בהינתן המיקום הנוכחי של פקמן, הרוח תחשב את המרחק (manhattan distance) מכל אחד מהמצבים אליהם היא יכולה להגיע – למיקום הנוכחי של פקמן. כעת, אם הרוח נמצאת במצב של ״מפחדת״ (התשובה תהיה כן אם הטיימר של מצב ״מפחדת״ טרם נגמר, כלומר הוא גדול מ-0, ולא אחרת), אזי ה-bestScore (נסביר בהמשך מה הוא) יהיה המרחק המקסימלי מפקמן, ואם היא אינה במצב של ״מפחדת״ אזי ה-bestScore יהיה המרחק המינימלי מפקמן. בנוסף נשמר משתנה בשם bestProb שהוא נקבע על ידי אחד מהפרמטרים prob\_attack ו-prob\_scaredFlee שמועברים לפונ׳, והוא קובע את גודל ההבדל ההבדל בהסתברויות בין מצב של ״בריחה״ למצב של ״תקיפה״.

לסיום, ה-bestProb מתחלק שווה בשווה בין כל הפעולות שהוגדרו באלגוריתם להיות הטובות ביותר, ואילו המשלים של bestProb (שהוא 1-bestProb) מתחלק שווה בשווה בין כל הפעולות החוקיות הבאות. לבסוף וקטור ההסתברויות מנורמל כך שסכומם יהיה 1.

כך נוצר מצב שבו, כתלות בפרמטרים המועברים לפונ׳ כמובן, פעולות טובות יותר מבחינת ״בריחה״/״תקיפה״ – כתלות במצב הרוחות - יועדפו על פני פעולות אחרות.

כלומר התפלגות התנועה של הרוח DirectionalGhost תהיה כתלות במה מצבה של הרוח. המצב הראשון הוא כאשר הרוח נמצאת במצב של ״מפחדת״, שהוא המצב לאחר שפקמן אכל קפסולה (ואז לפרק זמן מסוים הרוחות נצבעות בלבן והן במצב אכיל), ואז ההתפלגות תהיה שמצבים אשר ״בורחים״ מפקמן יועדפו, והמצב השני הוא המצב הדיפולטי כאשר הרוח נמצאת במצב ״רגיל״, שהוא המצב כאשר פקמן לא אכל קפסולה (ואז הרוחות הן בצבעים רגילים שאינם לבן והן אינן במצב אכיל), ואז ההתפלגות תהיה שמצבים אשר ״תוקפים״ את פקמן יועדפו.

דוגמה לפעולה של האלגוריתם: אם יש לרוח 4 פעולות חוקיות מהמצב הנוכחי, כאשר שתיים מהן עדיפות, והפרמטרים שהתקבלו הם prob\_attack = 0.8 ו-prob\_scaredFlee = 0.8 (הערכים הדיפולטיים לפונ׳), אזי 0.8 יתחלק שווה בשווה בין שתי הפעולות העדיפות, כלומר כל פעולה תקבל ראשית 0.4, וכמו-כן המשלים שהוא 0.2 יתחלק שווה בשווה בין על 4 הפעולות, כלומר כל פעולה תקבל תוספת של 0.05. לסיכום כל אחת משתי הפעולות העדיפות יהיו בהסתברות 0.45 וכל אחת משתי הפעולות האחרות יהיו בהסתברות של 0.05, והרוח תעדיף את הפעולות הטובות יותר מבחינתה.

1. מימוש בקוד.
2. למעשה המימוש של שני ה-Agents הוא זהה לגמרי למעט הבדל אחד. נסביר את החלק במימושים האחראי על ההתפלגויות:

על-מנת לממש את האלגוריתם של Expectimax, כפי שנלמד בהרצאות, אנו זקוקים למערך ההסתברויות התואם למערך הצעדים החוקיים האפשריים לביצוע. ואז כאשר מדובר במצב הסתברותי (במקרה שלנו – כאשר תורה של רוח לשחק) – אנו מעוניינים להחזיר את הסכום על הערכים שנסמנם b (מספר הגורמים בסכום הוא כמספר הצעדים החוקיים) כאשר b הוא מכפלת ערך ה-expectimax של צעד מסוים בהסתברות שהרוח תבחר בצעד זה. ולכן גם במימוש של RandomExpectimaxAgent (בסעיף ה) וגם במימוש של DirectionalExpectimaxAgent (בסעיף ו׳) אנו למעשים מעוניינים להשתמש בפונ׳ getDistribution שכבר מומשה עבורנו במחלקות של RandomGhost ו-DirectionalGhost בהתאמה. פונ׳ זו מספקת עבורנו את מערך ההסתברויות (התואם למערך הצעדים החוקיים האפשריים לביצוע) – כפי שאנו רוצים.

מכאן שההבדל במימושים הוא על איזה אובייקט אנו קוראים לפונ׳ getDistribution – עבור ה-RandomExpectimaxAgent אנחנו מייצרים אובייקט של RandomGhost ועבור ה-DirectionalExpectimaxAgent אנחנו מייצרים אובייקט של DirectionalGhost. בשני המקרים אנחנו מספקים לבנאי את האינדקס של הרוח ואז קוראים ל-getDistribution עליו, ומקבלים התפלגות שווה במקרה הראשון וכזאת שמעדיפה צעדים טובים יותר במקרה השני – בהתאמה.

1. רעיונות לשיפור אסטרטגית הרוחות:
   1. רוחות המשתפות פעולה בינן לבין עצמן – עד כה נתקלנו בקוד של רוח (RandomGhost ו-DirectionalGhost) ש­מבחינת מיקומים של ישויות אחרות בלוח מתחשב במיקום של פקמן ובמיקום של הרוח עצמה בלבד. מה שנוכל לעשות כדי לשפר את אסטרטגית הרוחות הוא שרוח בבואה לבחור את הצעד הבא לחישוב תסתכל גם על מיקום שאר הרוחות במשחק. אם לדוגמה רוח תבחין כי הרוח השנייה מתקרבת לפקמן מכיוון מזרח, ותבחין כי במסלול BFS לפקמן יש לו דרך מילוט מכיוון דרום, היא תוכל להגיע אליו מכיוון דרום ובכך הסיכוי שאחת מהרוחות ״תסגור עליו״ – גדל משמעותית. דבר זה לא היה מתאפשר ביחס לאסטרטגיות בסעיפים הקודמים מכיוון שכאמור רוח לא מתחשבת במיקום הרוחות האחרות, והסיכוי שהיא הייתה מגיעה ״לסגור״ על פקמן מאותו כיוון של הרוח האחרת – הוא גדול יותר בסעיפים הקודמים.

רעיון זה עשוי לשפר את אסטרטגית הרוחות משום שבמימושו אנו מבצעים שינוי קונספטואלי של הגישה – אם נסתכל על המשחק בכללותו, יש כאן צד אחד נגד צד שני – פקמן נגד הרוחות. אבל, בתוך הצד של הרוחות (כתלות בהגדרות המשחק) ישנן מספר רוחות, שלהן מטרה זהה. ומכיוון שבאסטרטגיה החדשה הרוחות הללו ישתפו פעולה כפי שהוסבר לעיל, אזי במקום שכל רוח תתרום לנצחון קבוצתה בצורה בלתי תלויה ברוחות האחרות, כעת הן יוכלו ״לתאם״ את הצעדים הבאים ביניהן וכך לשחק בצורה חכמה יותר – כקבוצה – דבר שיוביל ליותר נצחונות מבחינתם.

* 1. רוח שנותרת במקומה – ניתן לממש אסטרטגיה שבמהלכה בתחילת המשחק רוח תנוע לכיוון קפסולה וכל עוד פקמן לא קרוב – היא תישאר בה כאשר היא תבצע פעולות הלוח ושוב כדי להישאר באזור הקפסולה תמיד (״אזור״ – הכוונה המשבצת של הקפסולה ומשבצות שכנות לה).

רעיון זה עשוי לשפר את אסטרטגית הרוחות משום שהוא יצור מצב שבו פקמן יתקשה לסיים את המשחק, שכן הוא זקוק לקפסולה זו כדי לנצח, וככל שפקמן מתעכב ואינו אוכל קפסולה זו – הוא מפסיד נקודות (לפי הגדרת המשחק). בינתיים רוחות אחרות ינועו לכיוון פקמן כדי להבריח אותו מהרוח שנותרת במקומה ובשאיפה גם ״לאכול״ אותו. ואז אם הרוחות לא יצליחו בסופו של דבר להכניע את פקמן, הן לכל הפחות יעכבו אותו מאד בדרך אל הניצחון, וזה מהווה שיפור באסטרטגיה שלהן.

נדגיש כי אין הכוונה שהרוח תהיה תמיד סטטית אלא רק כאשר פקמן אינו קרוב אליה. אם פקמן קרוב אליה ויש לה אפשרות לבצע מהלך שיוביל לאכילתו – זה מה שהיא תעשה. בכך תגדיל את מספר ההפסדים של פקמן.

* 1. תזוזת רוחות כמקשה אחת – רעיון נוסף לשיפור אסטרטגית הרוחות, שהוא תלוי במבנה הלוח ומתאים בעיקר ללוחות שבהם יש כמות קטנה יחסית של קירות. במסגרת רעיון זה הרוחות ינועו כמקשה אחת – כלומר תמיד ינועו כאשר הן צמודות זו לזו, ובשאיפה כאשר הן פרוסות לאורך או לרוחב (ולא מקובצים יחדיו).

רעיון זה עשוי לשפר את אסטרטגית הרוחות במצבים מסוימים. אם נסתכל לדוגמה על לוח שבו אין קירות בכלל, ומספר גדול של רוחות, בהפעלת אסטרטגיה זו הן ינועו ביחד לכיוונו של פקמן, ומכיוון שהן נעות כמעין ״טור״ אורכי או ״שורה״ רוחבית בכיוון המתאים – מספר הצעדים שאותם פקמן יכול לבצע כדי להתחמק יורד משמעותית. הדבר נכון במיוחד כאשר מדובר על תזוזה של הרוחות כמקשה אחת לכיוון פינה כאשר פקמן נלכד בפינה.

הערה: הסיבה שרעיון זה מתאים ללוחות שבהם יש כמות קטנה יחסית של קירות היא שכאשר יש מעט קירות הרוחות יכולות לנוע כמקשה אחת ללא הפרעה, בעוד שכאשר ישנה כמות גדולה יחסית של קירות – הקירות יפריעו להן לשמור על סדר המקשה והרוחות יאלצו להסתדר מחדש אחת לכמה זמן.

הערה נוספת: הרוחות צריכות גם להפעיל שיקול דעת בנוסף לתזוזה כמקשה, כחלק מהאסטרטגיה החדשה, ואז כאשר פקמן קרוב מאד לרוח מסוימת והיא יכולה ״לאכול״ אותו רק אם תצא מהצורה של המקשה – ״אכילת״ פקמן נמצאת כמובן בעדיפות עליונה ועליה לבצע זאת.

חלק ז׳ – ניסוח השערות במשחק פקמן

השערת האפס (H0):

שימוש בסוכן MinimaxAgent עם עומק 4 מוביל את פקמן למספר ניצחונות גדול יותר מאשר שימוש בסוכן ReflexAgent (עם הפונ׳ היוריסטית הלא-משופרת)

השערה חלופית (H1):

שימוש בסוכן ReflexAgent (עם הפונ׳ היוריסטית הלא-משופרת) מוביל את פקמן למספר ניצחונות שווה או גדול יותר מאשר שימוש בסוכן MinimaxAgent עם עומק 4

המבחן הסטטיסטי:

נריץ כל אחד מהסוכנים 100 הרצות על כל אחד מ-10 הלוחות, בסה״כ 1000 הרצות לכל סוכן, ונבדוק את אחוז ההצלחה הממוצע של כל סוכן.

נדחה את השערת האפס: אם אחוזי ההצלחה של הסוכן MinimaxAgent גבוהים מאחוזי ההצלחה של הסוכן ReflexAgent בפחות מ-50% (או נמוכים מהם כמובן).

לא נדחה את השערת האפס: אחרת.

הסבר:

כפי שנלמד, השערת האפס היא ההשערה השמרנית, האמונה הרווחת. לעומתה ההשערה החלופית היא ההשערה החדשנית, המהפכנית. הסוכן ReflexAgent משתמש בעומק 1 והיוריסטיקה שבה הוא משתמש מתבססת על הניקוד שלו לאחר ביצוע הצעד המיועד. לעומת זאת, הסוכן MinimaxAgent משתמש בעומק 4 בהגדרה שלנו כאן, והיוריסטיקה שבה הוא משתמש מתבססת על שיפורים מהיוריסטיקה הפשוטה, שבין היתר כוללת בריחה מרוחות.

מכאן שהשערת האפס היא שהסוכן Minimax שגם חוקר יותר צעדים קדימה מהסוכן ReflexAgent וגם משתמש ביוריסטיקה טובה יותר מהסוכן ReflexAgent שמונעת ממנו להפסיד במקרים אחדים – יוביל את פקמן למספר ניצחונות גדול יותר.

בהתייחס למבחן ה-t-test, אנו למעשה רוצים לנסח את המבחן כך שרמת המוהבקות שלו תהיה מקסימלית. בחרנו בהפרש – 50 אחוזים – שאינו משאיר הרבה מקום לטעויות סטטיסטיות. דהיינו, אם היינו מקבלים שהסוכן MinimaxAgent הוביל את פקמן לנצחונות ב-5% יותר מהסוכן ReflexAgent וקובעים שבשל כך הסוכן MinimaxAgent הוא באופן מוחלט טוב מהסוכן ReflexAgent ומוביל את פקמן ליותר נצחונות באופן גורף – הרי שהסיכוי שמדובר בטעות סטטיסטית בתוצאה שלנו הוא גבוה מהסיכוי שישנה טעות שכזאת כאשר משתמשים בניסוח שלנו. וזאת בדיוק מהות ה-t-test.

חלק ח׳ – ניסויים, תוצאות ומסקנות

1. הקובץ נכלל בהגשה.
2. ששש
3. ששש
4. ששש
5. ששש
6. ששש

|  |  |  |
| --- | --- | --- |
| **DirectionalExpectimaxAgent** | **RandomExpectimaxAgent** |  |
| Avg. Score:  Win Rate: | Avg. Score:  Win Rate: | **RandomGhost** |
| Avg. Score:  Win Rate: | Avg. Score:  Win Rate: | **DirectionalGhost** |

1. ששש
2. ששש
3. ששש

חלק ט׳ – תחרות בקורס

מימוש בקוד.