פרויקט סיום בקורס מבוא לבינה מלאכותית

3D multi-player connect 4

מגישים:

איתמר שרם, 206762551

שלום בלוי, 319144762

עבד נירוך, 213668700

כאן יהיה תוכן עניינים

מבוא

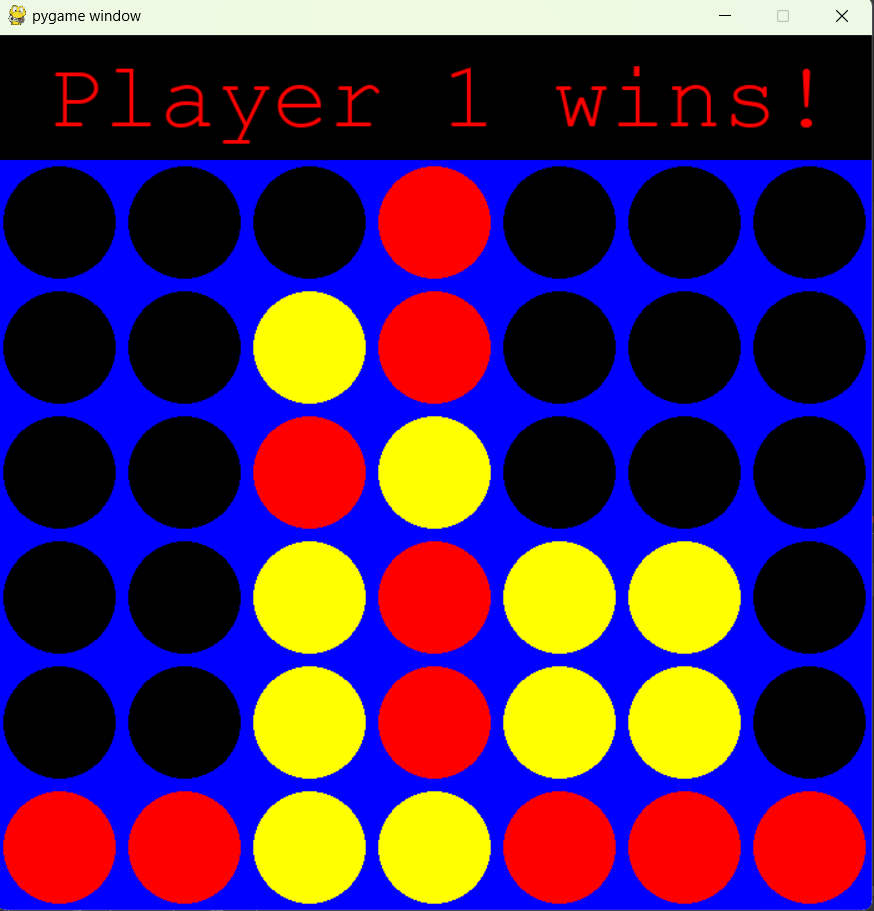
עבודות קודמות

מתודולוגיה

תוצאות

סיכום

ביביליוגרפיה(לא דרשו, אבל כדאי כדי לשים גם קישורים לUI ולעבודות קודמות וכו')

מבוא:

המשחק שלנו מבוסס על המשחק 4 בשורה. המשחק המקורי מתקיים בלוח דו ממדי בגודל קבוע, לשני שחקנים. כל שחקן בתורו משחיל דיסקית בצבע שלו לאחת העמודות בלוח. מטרתו של כל שחקן להגיע ראשון לרצף של 4 דיסקיות בצבע שלו. הרצף יכול להיות בשורה, בעמודה או באלכסון.

בפרויקט שלנו הרחבנו את המשחק, על מנת לאתגר את הסוכנים שלנו ולבחון אותם בסביבות מורכבות יותר. השדרוג חל בכמה אופנים:

* לוח המשחק מוגדר כעת ע"י הרביעייה הבאה: (R,C,D,W), כאשר R הוא מספר השורות, C הוא מספר העמודות, D הוא העומק וW הוא רצף הנצחון
* ניתן להריץ את המשחק עם יותר משני שחקנים, כל אחד מתחרה בשאר השחקנים על מנת לנצח.

עץ המצבים של המשחק גדל בצורה אקספוננציאלית- כאשר t הוא מספר התורות עד לניצחון. כלומר סוכנים שיממשו אלגוריתמי חיפוש פשוטים ירוצו למשך זמן רב! לפיכך צריך לחשוב על אלגוריתמים חכמים, וזו הסיבה שבחרנו במשחק זה.

לקחנו מימוש למשחק 4 בשורה שמצאנו באינטרנט, על מנת לאפשר לבן אדם לשחק נגד אחד הסוכנים בצורה אינטרקטיבית. שדרגנו משמעותית מימוש זה כדי שיתאים להרחבות שהוספנו למשחק. להוסיף קישור לgithub של המימוש, או רפרנס לנקודה מתאימה בביביליוגרפיה.

מימשנו סוכני AI שונים, לפי רעיונות שונים שלמדנו בכיתה, שמטרתם היא לנצח במשחק את השחקנים שיתמודדו מולם. הסוכנים שמימשנו הם:

1. minmax\_agent
2. alpha\_beta\_agent
3. Q-learning\_agent
4. רשת CNN

נשים לב שהמשחק המקורי הוא משחק סכום אפס- נצחונו של שחקן אחד הוא הפסדו של האחר. לפיכך חשבנו כי האלגוריתם הטבעי ביותר להתחיל איתו הוא אלגוריתם המינמקס, שמותאם לנצח בסוג זה של משחקים. אלגוריתם מינמקס במימוש נאיבי, יורד עד לעומק עץ המצבים, ומהעלים מתחיל לשערך את הציון למצבים. במשחק 4 בשורה, שבו עץ המצבים עמוק מאוד, דבר זה אינו ישים. לכן נצטרך להגביל את עומק החיפוש שלנו ולתת ציון למצב לוח למרות שהמשחק עדיין לא הסתיים. לשם כך השתמשנו בהיוריסטיקה/היוריסטיקות שהגדרנו, אותן נתאר בחלק המתודולוגיה.

סוכן האלפא-בטא הוא הסוכן ההגיוני הבא- הוא פועל כמו סוכן המינמקס מבחינת ההחלטות שבוחר במהלך המשחק, אך מדלג על ענפים מסוימים ובכך מקצר את זמן הריצה.

לסיום, רצינו לבחון את ביצועיו של סוכן reinforcement learning . דרך אפשרית להפעיל את האלגוריתם במקרה שלנו, היא לתת לשחקן למידת החיזוק לשחק מול שחקן מינימקס/רנדומי ודרכו ללמוד איך לשחק את המשחק בצורה הטובה ביותר.

סוכן reinforcement learning – סוכן זה לומד באמצעות משחקים רבים את הפעולות הכדאיות בהינתן מצבי הלוח, כאשר במהלך הלמידה הסוכן מעדכן את הציון על מצב לוח מסוים, באמצעות הרצת סימולציות עד לנצחון/הפסד(monte carlo).

אולי יהיה הסבר קצר על רשת CNN

עבודות קודמות:

בחנו עבודות קודמות(לשים קישורים לרלוונטיים בביביליוגרפיה)

בשתיהן, האלגוריתמים בהם השתמשו הם minmax ו-alpha beta pruning . כפי שאנחנו הסקנו, עץ המצבים במשחק גדול מאוד, ולכן החוקרים(?) בחרו להגביל את עומק החיפוש ולחשב ציון ללוח של משחק שטרם הסתיים. ההבדלים בין האלגוריתמים שהם בחרו הוא בפונקציית האבליואציה שמחשבת ציון ללוח. נציג את ההיוריסטיקות המוצלחות ביותר בכל אחת מהעבודות.

נסמן ב-N את אורך הרצף המוגדר עבור נצחון במשחק(ברירת המחדל היא 4)

פונקציה א:

במשחק 4 בשורה עם לוח דיפולטיבי(6 שורות על 7 עמודות) ישנן 69 פוזיציות בהן אפשר להגיע לנצחון(24 אופקיות, 21 אנכיות ו24 אלכסוניות). אם ברצף באורך N יש לשני השחקנים דיסקיות, נתעלם מרצף זה(שכן אף אחד מהם לא יכול לנצח שם). אחרת, לכל רצף בלוח ניתן ערך לפי כמות הדיסקיות שיש בו לשחקן מסוים. נסכום את כל הערכים הללו עבור השחקן הראשי, ונחסר מהם את סכום הערכים של היריב.

בהיוריסטיקה זו יש כמה חסרונות:

* הערכים שנבחרו מותאמים עבור לוח בגודל הדיפולטיבי בלבד, וקשה להכליל אותה למשחק עם לוח בגודל שונה, עם רצף נצחון שונה.
* ההיוריסטיקה יכולה לתת ערך זהה ללוח בו יש לנו הרבה רצפים קצרים, לעומת מעט רצפים ארוכים. אם נחשוב על הרלקסציה שבה ככל שיש לשחקן יותר דיסקיות ברצף, הוא יותר קרוב לנצחון, היוריסטיקה זו מתעלמת מעיקרון זה.

פונקציה ב(נקראה IBEF2): נספור רצפים לא חסומים(שלא מכילים דיסקיות של היריב).ונחזיר , עבור k מספר הדיסקיות ברצף זה. הפונקציה תסכום את הערכים הללו ותחסיר מערך זה את סכום הערכים של היריב.

חסרונות של היוריסטיקה זו:

בדומה להיוריסטיקה הקודמת, יכולה לתת ערך זהה ללוח בו יש לנו הרבה רצפים קצרים, לעומת מעט רצפים ארוכים.

מתודולוגיה:

מינמקס, אלפאבטא והיוריסטיקות

הרחבת אלגוריתמי minmax,alphabeta עבור משחק של יותר משני שחקנים:

במקרה זה, השחקן הראשי יהיה שחקן המקסימום, ומבחינתו כל שאר השחקנים יהיו שחקני המינימום. בשערוך הערך עבור כל קודקוד בעץ ששחקן המקסימום מגדיר(עד עומק מסוים), שחקן המקסימום יבחר את הערך המקסימלי מבין ערכי כל ילדיו. כל אחד משחקני המינימום יבחר את המינימום מבין ערכי כל ילדיו.

נשים לב כי בגישה זו, מטרתו של כל שחקן מינימום היא לוודא ששחקן המקסימום לא מנצח, גם אם זה גורר ששחקן מינימום אחר ינצח- מכיוון שההיוריסטיקות מחושבות במינוס על כלל היריבים, אזי מטרת כל אחד משחקני המינימום היא שכל השחקנים היריבים(כולל הם עצמם) יקבלו ניקוד גבוה, וכך יורידו מערכו של שחקן המקסימום. בעיני שחקן המקסימום, כלל יריביו עשו יד אחת נגדו.

מימשנו מספר היוריסטיקות פשוטות לצורך בדיקת שאר השחקנים, עליהם נפרט בשלב התוצאות. כעת נציג את ההיוריסטיקה המרכזית ששימשה את אלגוריתם המינמקס:

היוריסטיקת complex (לשנות בקוד בarg\_parse לcomplex במקום allcomplex)

ההיוריסטיקה פועלת באופן הבא:

בהינתן לוח מסוים, ההיוריסטיקה תבחן את כל הרצפים השונים הקיימים על הלוח אשר לא חסומים, לכל לוח היא מחזירה שני ערכים, שמייצגים את הרצף הארוך ביותר שהושג בלוח, וכמות המופעים שלו. מאלו מחסרים את שקלול הערכים המתאימים ליריבים באופן הבא:

* מהרצף הארוך ביותר של שחקן המקסימום, נחסיר את ממוצע הרצפים המקסימליים של היריבים
* מכמות הרצפים הארוכים ביותר של השחקן המקסימלי, נחסיר רק את כמויות הרצפים המקסימליים של יריבים שהשיגו אותו רצף מקסימלי זהה לשל שחקן המקסימום- אנחנו לא רוצים לתת משקל ליריבים "חלשים" יותר בלוח נתון. רק יריבים שמאיימים על השחקן שלנו באים לידי ביטוי בציון הלוח.

לדוגמא:  
רצף הנצחון מוגדר להיות 4, שחקן המינמקס הוא השחקן הורוד. הרצף הכי ארוך שלו הוא 3, וכמות הרצפים באורך זה היא 1. לאדום יש רצף אחד באורך 3 (הרצף השני שלו באורך 3 חסום). לצהוב חמישה רצפים באורך 1 (ישנם רצפים שנספרים יותר מפעם אחת, עבור כל רצף נצחון בו הם יכולים להשתתף). ולכן פונקציית ההיוריסטיקה תחזיר את הציון הבא:

נשים לב כי בדרך זו, אנו נותנים עדיפות לאורך הרצף, ורק כאשר הרצפים זהים אנחנו מסתכלים על מספר המופעים. בכך אנו מתגברים על הבעיה שהצגנו בהיוריסטיקות מהעבודות הקודמות.

כאשר הרחבנו את המשחק ללוח תלת ממדי, נתקלנו בבעיה- על מנת לבדוק האם הנחה של דיסקית הובילה לנצחון, עלינו לבדוק רצפים רבים שיכולים להביא לנצחון. דבר זה פגע בזמני הריצה. לכן על מנת לשפר את זמני הריצה, אנחנו מחזיקים עבור כל לוח את מפת כל הרצפים המופיעים בו, וכאשר מוסיפים דיסקית, מעדכנים את מפות הרצפים רק באותו אזור מצומצם. הדבר שיפר משמעותית את זמני הריצה (למשל כדי למצוא את הרצף הארוך ביותר בלוח, מסתכלים במפת הרצפים)

הסוכן הכפול

בarg parse להוציא את האפשרות לבחור בrl\_agent , ולשנות אתcnn\_agent להיקרא double\_agent

כדי שסוכן q-learning יציג תוצאות טובות, עליו להכיר הרבה מצבים במשחק, על מנת שידע לפעול בכל סיטואציה שייתקל בה. יתרה מזאת, אם יפגוש מצב שלא ראה קודם לכן, אפילו אם ראה מצב דומה(למשל תבניות דומות, אבל בהזזה מהמצב שכן ראה) הוא לא ידע כיצד לפעול.

לפיכך, היינו רוצים מרחב מצבים קטן ככל האפשר. מצד שני, ככל שייצוג המצב פשוט יותר, יותר מידע הולך לאיבוד.

ניתן היה לבחור ייצוג שמחזיק מידע חלקי לגבי הלוח, אך בחרנו להתמודד עם בעיה זו באופן הבא:

במשחק שלנו ישנה חשיבות רבה לתבניות, שיכולות להופיע במיקומים שונים בלוח, ולכן כדאי להשתמש בכלי שהוא translation invariance , שמסוגל לזהות תבנית ללא תלות במיקום. בכך, נתפוס את הטוב משני העולמות- מצד אחד נחזיק ייצוג מלא של הלוח, ומצד שני נוכל לזהות תבניות בלוחות שראינו על מנת להכליל ללוחות דומים.

לפיכך יצרנו את הסוכן הכפול- סוכן זה משתמש בסוכן Q-learning שילמד כמה שיותר מצבי לוח. בשל החסרון המוטמע שאינו יכול להכליל לוחות אלה, ישתמש בסוכן CNN, שהתאמן על הלוחות הנ"ל, ידע להכליל עליהן, ויבחר עבור הסוכן שלנו את המהלך הטוב ביותר.

סוכן Q-Learning

פונקציית הרווח:

כפי שלמדנו בקורס, על מנת שסוכן הRL ילמד לשחק במשחק, עלינו לספק פונקציית רווח שתיתן לו ציון על הפעולה שביצע. הפונקציה שבחרנו פועלת באופן הבא: ראשית תתגמל אותו על נצחון. אם לא ניצח אך הגענו לתיקו, תיתן ציון חיובי קטן יותר. אם הוא הולך להפסיד בתור הבא לאחד השחקנים, ניתן ציון שלילי, שערכו בערך מוחלט קטן ככל שהשחקן שרד למשך יותר משחקים.

מאופן פעולת האלגוריתם, ערכים אלה יפעפעו מטה עבור לוחות שרחוקים מהכרעה, ולשם כך דרושים משחקים רבים לצורך אימון הסוכן.

אופן האימון

אימנו סוכן זה לשחק מול \_\_\_\_\_ , מכיוון שאנו רוצים שיראה \_\_\_\_\_\_ . בפונקציית הרווח, כאשר נרצה לסמלץ מהלך של יריב, נסמלץ מהלך של יריב ה\_\_\_\_\_\_, כי אנחנו רוצים \_\_\_\_\_\_ .

סוכן CNN

פרסור ה-Q table

כזכור, סוכן ה-Q-learning מתחזק טבלה, שבה עבור כל מצב ופעולה בהם נתקל, קיים ציון. המרנו טבלה זו, למאגר מידע, בו לכל לוח יש ציון. הדרך שבה השגנו את הציון הזה, היא ע"י מעבר על הQ-table , לכל מצב עם ערך שאינו טריוואלי (שמעיד על כך שמידע לא פעפע לנקודה זו), יפעיל את הפעולה על הלוח, ונקבל לוח חדש, עם ציון זהה. נמצע ציונים של לוחות זהים. כלומר הדגימות שאנו מעבירים לרשת הם לוח, וציונו.

רשת ה-CNN

רשת זו תהיה מורכבת מקונבולוציות, עם קרנלים בגודל רצף הנצחון. הקרנלים יסרקו את לוחות המשחק שיקבלו, ונצפה שכל קרנל יחפש תבניות אחרות בלוח.

לאחר שכבות הקונבולוציה, נפעיל שכבות לינאריות שישקללו את כל הנתונים שאספו הקרנלים לצורך חיזוי ערך הלוח.

בחירת פעולה

בזמן משחק, כאשר סוכן ה-CNN ירצה לבצע פעולה, יחשב את כל הלוחות הישיגים ע"י הפעולות החוקיות, יפעיל על כל אחד מהם את הרשת, ויבחר בפעולה שהובילה ללוח שקיבל את הציון הגבוה ביותר.

תוצאות

ביביליוגרפיה:

* <https://www.cs.huji.ac.il/course/2021/ai/projects/old/4InRow_2.pdf>
* <https://www.cs.huji.ac.il/course/2021/ai/projects/old/4InRow_1.pdf>
* <https://en.wikipedia.org/wiki/Connect_Four>
* <https://www.fierz.ch/strategy1.htm>

דברים להוסיף ולערוך:

1. לעבד- צריך לשנות את ההיוריסטיקה IBEF2 כך שהיא באמת תהיה דומה לה.
2. מעבר לרשת CNN: צריך ליצור דאטה(נריץ מונטה קרלו מספר רב של פעמים ממצבי לוח, על מנת לקבל עבורם ערך. זה יהיה הלייבל של הלוח, שרשת הנוירונים תיתקל בה במהלך האימון.   
   נאמן רשת CNN שתקבל את הדאטה סט הנ"ל, ותפיק עבור כל לוח כזה ציון. סוכן הCNN בהנתן מצב לוח כלשהו, ישחק את כל המהלכים האפשריים עבורו, ויבחר במהלך שנותן לוח עם ציון מקסימלי.
3. לחשוב האם ניתן לחשוב על פונקציית היוריסטיקה אחרת, שבה באלגוריתם minmax עם כמה משתתפים, שחקן המקסימום לא יחשוב שכולם נגדו, אלא כולם נגד כולם.