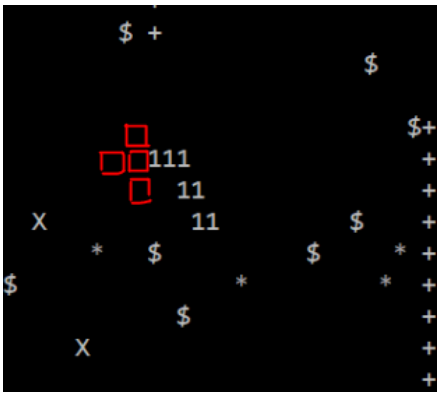
אור קוסטה 20402248
נועם קפלן 308373505

APML Snake Project

1. הצגת ה state :

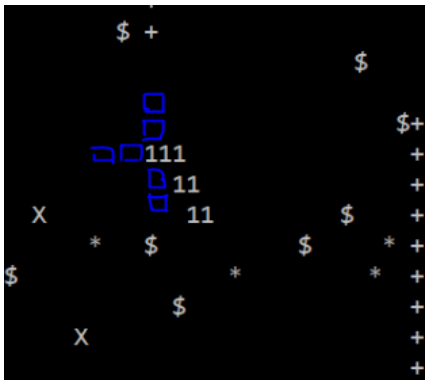
עבור המודל הלינארי הייצוג הוא בווקטור הכולל מידע על 4 נקודות בלוח, הנקודה אליה נגיע כשנבצע את הפעולה x ועוד שלושה צעדים, קדימה ימינה ושמאלה מנקודה זאת.



למשל עבור הפעולה F הווקטור יכיל את המשבצות המסומנות בתמונה כל אחת מהמשבצות מיוצגת בווקטור בצורה בינארית כלומר כל נקודה היא וקטור בגודל 11 (עבור 10 ערכים אפשריים מ 1- ל 9) של אפסים והערך 1 באינדקס המתאים לערך.
בנוסף הוספנו bias על לידי הוספת ערך נוסף שערכו תמיד 1 כדי לייצר משקולת חופשית.
סה"כ ווקטור הפיצ'רים במודל הלינארי הוא וקטור בגודל 45 של אפסים עם 4 ביטים דלוקים בהתאם לנקודות ועוד ביט דלוק לbias .

בחירת יותר נקודות מאשר ה4 הנ"ל לא הניבה תוצאה טובה יותר לכן בסופו של דבר נשארנו עם 4 בלבד. לגבי הייצוג הבינארי, תחילה נסינו לקחת פשוט ווקטור בגודל 4 עם הערכים, ראינו כי הלמידה לא כל כך פשוטה. זה היה הגיוני מבחינתנו להעביר לתצוגה בינארי שכן ערכים קרובים יכולים ליצור בלבול במשקולות במיוחד אם ההרצה הסופית כוללת משקלים רנדומליים ואז הפרס הטוב יכול להיות "6" בעוד הפרס הלא טוב יכול להיות "5". לאחר המעבר לבינארי אופן הלמידה השתנה לטובה לכן השארנו אותו כך.
בהמשך לידע שצברנו בקורס וגם מקורסים אחרים אנו יודעים שהוספת bias צריכה לתת עוד גמישות ויכולות מסוימת למודל. הוספנו עוד כניסה בווקטור לטובת ה bias ובממוצע נראה שזה הועיל סה"כ.

עבור המודל השני, ייצוג הstate הוא על ידי התבוננות על 6 מצבים קרובים, שניים לכל כיוון כפי שמסומן בתמונה. בדומה למודל הלינארי הייצוג גם כאן בינארי, ווקטור בגודל 66.



את הייצוג הבינארי בחרנו מאותם שיקולים שרשמנו קודם, כמו כן נרצה לציין כי גם כאן עבדנו תחילה עם ווקטור בייצוג על פי הערכים והתוצאות היו פחות טובות.
כמו כן, ביצענו ניסיונות להגדיל את כמות הנקודות על הלוח שאנחנו לוקחים, לקיחת צעדים קדימה יותר אמנם הכבידה מעט על הרשת אבל לא באמת עזרה לשפר ואפילו הייתה פחות טובה בכמה מקרים (יכול להיות גם שלא הצלחנו לייצר לומד טוב לכמות משקולות כאלה).
לכן נשארנו בסוף עם 6 משבצות מהלוח.

2. החלטנו להשתמש ברשת נוירונים לבניית הפונקציה Q. הסיבה בהתאם למה שלמדנו, בבעיה הזאת לא ניתן להציג את כל הלוח לבעיה ולכן יש צורך לבצע קירוב על ידי בחירה של כמות מידע מצומצמת יותר. ראינו את הרעיון בהרצאה וקראנו עליו עוד ברשת. ראינו כי יש גישה בה מכניסים את כל הלוח כ input ומבצעים רשת קונבולוציה, לא נסינו את זה אבל הסקנו שזה יכול להיות בעייתי מעט, כמו כן ראינו בהקלטות של המשחק שהלוח אחרי כמה זמן מתמלא בהרבה מידע והופך להיות עמוס, מה שיכול אולי להקשות על הלמידה בצורה הנ"ל (בנוסף היה רמז בתרגיל שזה מורכב). לכן החלטנו גם כאן בדומה למודל הלינארי (שעבד טוב באופן הזה) לקחת מידע על נקודות קרובות בלבד.

בנינו מודל המתבסס על רשת עם 3 שכבות FC בין שכבת ה input לשכבת ה output . המודל מקבל את הפיצ'רים כפי שהסברנו קודם ומחזיר וקטור בגודל 3 כאשר כל כניסה מתאימה לפעולה אפשרית. השכבה האחרונה עוברת אקטיבציה של softmax כדי להחזיר ערכים הסתברותיים ב output.
ביצענו ניסיונות להגדיל את הרשת אך זה לא עזר לנו להשתפר ואפילו הוביל לתוצאות פחות טובות, הגיוני סה"כ בהתחשבת ביחס שבין מורכבות הקלט שבחרנו למורכבות הרשת. ביצענו גם ניסיונות לרדת לשכבה אחת וזה הוביל אותנו לתוצאות שליליות. האופטימלי היה 3 שכבות (לא גדולות, 10,6,3).

3. המודל הלינארי:
השתמשנו באלגוריתם שלמדנו בהרצאה על q learning. בכל איטרציה בדקנו ייצוג אפשרי של כל מצב עבור כל פעולה אפשרית ושמרנו את הערך של Q עבור כל פעולה, לבסוף החזרנו את הפעולה עם הערך המקסימלי.
בתהליך הלמידה ביצענו את העדכון ל W ווקטור המשקולות בהתאם למה שלמדנו לפי הנוסחה :

$$\delta_{s_t,a_t}(\theta) = Q_\theta(s_t, a_t) - \left(r_t + \gamma \max_{a'} Q_{\theta_t}(s_{t+1}, a')\right)$$

$$\theta_{t+1} = \theta_t - \eta_t \delta_{s_t,a_t}(\theta_t)\, \nabla Q_\theta(s_t, a_t)$$

מודל ה custom :
השתמשנו באלגוריתם של הQ deep מההרצאה.
המודל למעשה מקבל וקטור המתאר קירוב של הלוח על ידי הצגת המשבצות סביב השחקן ומחזיר פרדיקציה לגבי הפעולה האופטימלית לביצוע. בכל שלב של act פשוט הרצנו את המודל הנוכחי והחזרנו את התוצאה שהוא נותן, בנוסף שמרנו והכנסנו את הנתונים ל queue שבתוכו שמרנו את ה"היסטוריה".
בכל שלב של למידה יש הסתמכות על פעולות היסטוריות, 5 סה"כ מתוך 1000 שאנחנו שומרים. בחרנו 5 לפי מספר הצעדים שעוברים בין כל קריאה לlearn ככה שבממוצע הלומד שלנו מתקדם ולומד בכל צעד. תהליך האופטימיזציה מובנה ומתבצע על ידי ספריית keras , בחרנו ספציפית לעבוד עם אופטימייזר Adam שעובד בדומה לSGD. את הloss הגדרנו לפי הq learning כפי שראינו בהרצאה לפי הנוסחה שהצגנו מקודם.

4. עבור exploration-exploitation בחרנו בהתאם למה שלמדנו בשיעור להתחיל עם ערך גבוהה של epsilon שהולך ודועך ככל שהמשחק מתקדם. בכל 100 איטרציות הכפלנו את epsilon ב 0.99 ווידאנו שהוא לא נמוך מ 0.05 .
עבור המקטע האחרון של המשחק החלטנו להוריד את epsilon להיות 0 אך בתצפיות על המשחקים ראינו כי קורה לפעמים שהנחש פשוט מסתובב סביב עצמו כשאין מידע רלוונטי קרוב אליו אז החלטנו גם במקטע האחרון להשאיר ערך גדול מ0. זה הכניס רנדומליות שלא כל כך אהבנו (לאחר צפייה במשחק) אז בגירסה האחרונה שהגשנו עשינו תיקון שאם יש שוויון בין המהלכים שהנחש יתקדם קדימה.
באופן כללי ראינו כי להתחיל מערך גדול של epsilon אכן תורם ללומד לעבוד טוב יותר בשלב הסופי של המשחק.
עבור המודל השני, custom, הרגשנו שיש יותר מידי רנדומליות שאולי מטעה את הרשת לכן החלטנו לבצע מנגנון בקוד שבכל 1000 איטרציות יכניס רצף של 100 פעולות שהן שייכות להחלטות הרשת בלבד (המטרה הייתה שהן ברצף ולא מופרעות על ידי פעולה רנדומלית). כיוון שאנו שומרים כל הזמן 100 צעדים אחורה זה גורם לכך ש 10 אחוזים מהדאטא מראה פעולות רצופות של הרשת. לא ראינו שיפור ניכר בזכות זה. בדיעבד הבנו שהשיפור היה נראה באופן ברור יותר בהרצות האימונים שביצענו שהיו מתחת ל 50 אלף איטרציות.

5. תוצאות:
המודל הלינארי:

python Snake.py -D 5000 -P Linear();Avoid(epsilon=0);Avoid(epsilon=0) -bs (20,60) -plt 0.05 -pat 0.01 -s 1000 -rt test2.pkl

מקבל ניקוד ממוצע של 0.4 ושני המודלים הנוספים מקבלים ניקוד ממוצע של 0.1

מודל ה custom :
D 50000 -P Custom();Avoid(epsilon=0);Avoid(epsilon=0);Avoid(epsilon=0);Avoid(epsilon=0) -bs (20,60) -plt 0.05 -pat 0.01 –s 5000

קיבלנו בקירוב 0.17 עבור הלומד שלנו ו 0.7 עבור ה Avoid הנוספים.

**גם בcluster קיבלנו תוצאות באותו סדר גודל.

6. מודלים נוספים:
ראינו כי באופן כללי הלומד הלינארי שלנו עובד כמו המודל העמוק ואפילו טיפה טוב יותר בממוצע. רצינו לחשוב על רעיון לשפר את המודל הלינארי עוד קצת. בנינו רשת נוירונים עם שכבת FC אחת שמקבלת את אותו הקלט שהשתמשנו עבורו במודל הלינארי ומוציאה ערך 1 שהוא בעצם ה q value הרשת שלנו בעצם מקבלת state ומחזירה הערכה כמה הוא שווה ולא action כמו במודל שעשינו. המטרה: לשכלל את המודל הלינראי שלנו אבל להישאר עם אותו אלגוריתם, רק במקום מכפלה פנימית רשת שפעולתה קרובה מאוד למכפלה פנימית ותהליך הלמידה שלה מתבסס על SGD של keras . בנוסף על ידי השימוש ברשת הייתה לנו פלטפורמה נוחה להשתמש בעיקרון של שמירת היסטוריה ואימון על 5 פעולות רנדומליות מהעבר בדומה למודל הdeep שהרכבנו.
תוצאות הרשת: עבור אותה הרצה של מודל ה custom מגיע ל 0.35 בממוצע.

סה"כ לא היה פה הרבה חידוש מבחינה אלגוריתמית אבל היה לנו נחמד לנסות ולראות. החלטנו בסוף להגיש את המודל הקלאסי של הרשת נוירונים שמקבל state שלם ומחזיר את הפעולה האופטימלית שהוא שונה בעיקרון שלו מהלינארי ועזר לנו ללמוד משהו חדש ומעניין.
כמו כן הגשנו גם את המודל ההיברידי הזה **בעיקר כדי שישתתף בתחרות הסופית** (אם זה אפשרי כמובן).
המודל מוגש תחת השם : policy_custom2