**פרויקט בקורס MAS - אליהו מינץ**

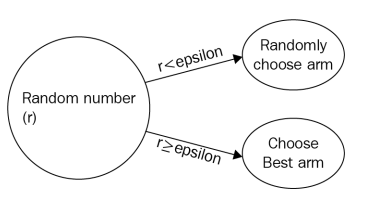
1. **מבוא:**

בפרויקט הנוכחי בחרתי לממש סוכנים אשר יכולים לשחק במשחק Multi-armed bandit (MAB). הפרויקט מכיל ארבעה קבצים. בקובץ bandits.py ממומשים סוגי בנדיטים שונים כאשר BanditEnv הבנוי בעזרת OpenAI gym environment מהווה Base class עבור בנדיטים אחרים (מימוש הושג ב-(github. בקובץ agents.py מימשתי שני סוגי סוכנים בעלי אסטרטגית משחק שונה. בקובץ compare\_agents.py מבוצעת השוואה של ביצועים של האלגוריתמים שנבחרו. בקובץ play.py ממומשת דוגמא של ההרצה של המשחק.

1. **תיאוריה**:

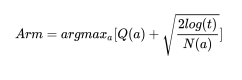
MAB נותן reward אקראי לפי התפלגות של הסתברות בה תוכנן מראש. המטרה של הסוכן (שחקן) למקסם את ה- total reward אחרי מספר משחקים מול כמה מכונות. הסוכן בכזה סוג של המשחק עומד בפני דילמה שלexploitation מול exploration. שני סוכנים מומשו בפרויקט עם שני אלגוריתמים המסוגלים להתמודד בסביבה זו: Epsilon Greedy ו- Upper Confidence Bound .

1. **שיטות**:
2. אלגוריתם ε-greedy מגריל כל פעם מספר בתחום בין 0 ל-1. במידה ומספר גדול מ- ε, הסוכן בוחר במכונה הטובה ביותר (מבחינת ה- reward המצטבר). אחרת נבחרת מכונה באופן אקראי. התהליך מתואר בתיאור 1.

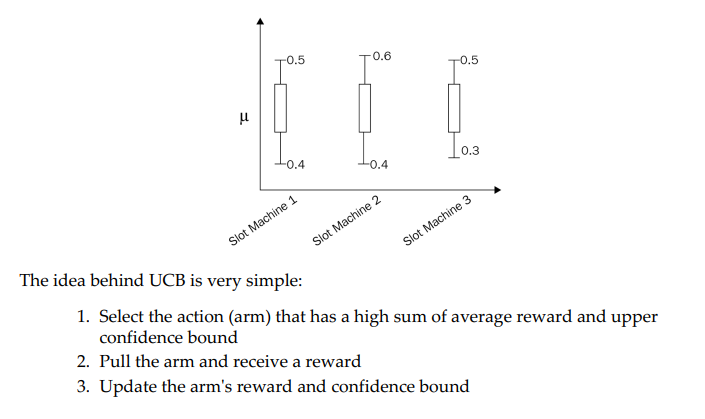


תיאור 1 : תיאור של אלגוריתם ε-greedy

1. אלגוריתם UCB מבצע exploration לפי מכונה שבה הממוצע של ה-reward המצטבר ו-UCB נותנים תוצאה הגדולה ביותר לפי נוסחה



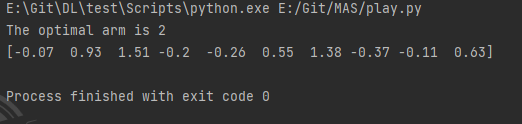
בתיאור 2 מתואר האלגוריתם באופן הבא:

תיאור 2: UCB algorithm

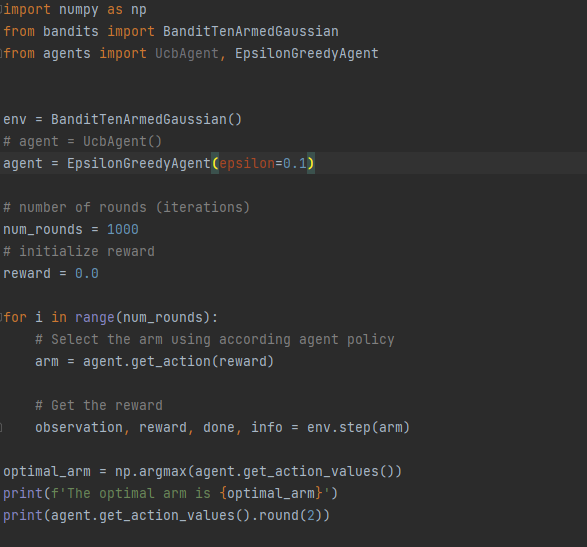
1. **תוצאות**:

בהרצות נבחר בנדיט עם 10 ידיים עם הסתברות נורמלית של ה-rewards.

ניתן לבחור בסוכן בעל אסטרטגיה של ε-greedy (EpsilonGreedyAgent) עם פרמטר שקובע את ה- ε או סוכן המשתמש באסטרטגיה של UCB (UcbAgent). התוכנית ההרצה בקובץ play.py מוצגת בתיאור 4. בכל פעם הסוכן מקבל reward מהריצה האחרונה ובוחר ע"י get\_action את המכונה בה ישחק במשחק הבא. בסוף ההרצה מודפסים כל ה-rewards שצבר הסוכן על כל המכונות והמכונה האופטימלית מבחינת התוצאה המצטברת:



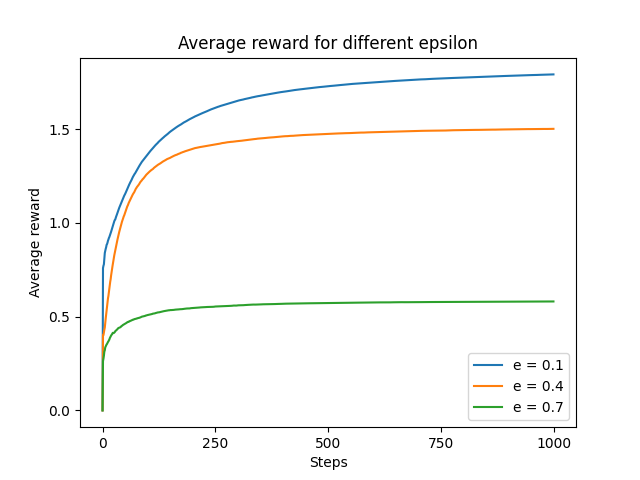
תיאור 3: תוצאות המשחק אחרי 1000 arm pulls



תיאור 4: תוכנת הרצה play.py

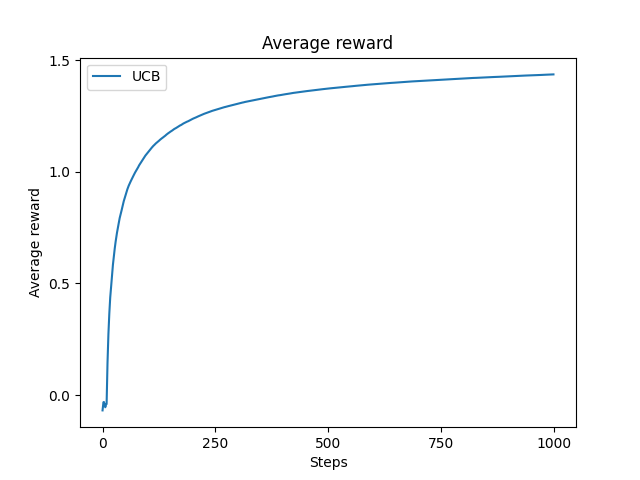
1. **ניתוח**:

ע"מ לבדוק השפעה של גודל ה- ε על average reward, ביצעתי הרצה של 1000 ניסויים של 1000 משחקים עבור שלושה ערכים שונים של ה- ε. תוצאות של הניסוי מוצגים בתיאור 5. רואים שהביצועים הטובים ביותר מושגים עבור ה- ε ששווה 0.1. בערך של 0.1 נמצא האיזון האופטימלי בין exploitation לבין exploration.



תיאור 5: תוצאות של average reward עבור ε שונים ((ε-greedy agent

כמו כן, אותה בדיקה הרצתי עבור UCB agent ותוצאה מוצגת בתיאור 6.



תיאור 6: תוצאות של average reward (UCB agent)

**נספח:**

רשימת הקבצים:

bandits.py

agents.py

play.py

compare\_agents.py

רשימת מקורות:

**"** Hands-On Reinforcement Learning with Python"

by Sudharsan Ravichandiran 2018 Packt Publishing