ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Reinforcement Learning and Dynamic Optimization
Project 2: Stock Market Trading --- Phase 2

Group 7

Όνομα: Κουτσοβασίλης Βασίλειος Α.Μ.: 2020030137

Όνομα: Τσικριτζάκης Γεώργιος-Μάριος Α.Μ. 2020030055

Task 1

Το πρώτο task του project αφορά την εύρεση της βέλτιστης πολιτικής που θα πρέπει να ακολουθήσει ένας οποιοσδήποτε agent, ώστε να μεγιστοποιήσει τα κέρδη του σε βάθος χρόνου, με βάση τον αλγόριθμο Tabular Q Learning. Για το συγκεκριμένο task χρησιμοποιήθηκαν τα δύο σενάρια των question 1 και question 2 που είχαν χρησιμοποιηθεί και στο assignment 3. Συγκεκριμένα:

Το **πρώτο σενάριο** περιγράφεται από το εξής περιβάλλον:

- Μετοχές: N = 2
- Transaction Fee: c = 0.8
- Discount Factor: γ = 0
- Rewards στο High State: r_H = [0.08, 0.04]
- Rewards στο Low State: r_L = [0.01, -0.02]
- Πιθανότητες Μετάβασης:
    - Από High σε High State: P_{HH} = [0.9, 0.85]
    - Από High σε Low State: P_{HL} = [0.1, 0.15]
    - Από Low σε High State: P_{LH} = [0.1, 0.2]
    - Από Low σε Low State: P_{LL} = [0.9, 0.8]

Λόγω της τιμής του γ (μας ενδιαφέρει η απόκτηση άμεσων κερδών) καθώς και της υψηλής τιμής του transaction fee σε σχέση με τις μέγιστες τιμές των reward, είναι προφανές ότι σε αυτό το περιβάλλον η βέλτιστη πολιτική θα είναι να διατηρεί ο agent την μετοχή με την οποία ξεκίνησε, δηλαδή να μην κάνει switch ποτέ.

Για τον αλγόριθμο Tabular Q Learning οι παράμετροι που τέθηκαν για να επιφέρουν την βέλτιστη πολιτική είναι:

- Episodes: 10000
- Learning Rate: α = 0.1 (δεν χρειάστηκε μείωση με την πάροδο του χρόνου)
- Αρχικό ε για την ε-Greedy επιλογή action: ε = 1.0
- Decay Factor του ε: ε_{decay} = 0.001
- Μέγιστος αριθμός βημάτων ανά επεισόδιο: max_steps_per_episode = 100

Για την εύρεση της βέλτιστης πολιτικής ο αλγόριθμος τρέχει πολλά επεισόδια και για κάθε επεισόδιο τρέχει έναν σχετικά μικρό αριθμό βημάτων που όταν συμπληρωθούν τερματίζει το κάθε επεισόδιο και γίνεται reset το περιβάλλον (αρχίζει ξανά από random state). Αυτό γίνεται γιατί εξορισμού δεν υπάρχει κάποια τερματική συνθήκη στο συγκεκριμένο πρόβλημα μετοχών και άρα ορίζοντας έναν μέγιστο αριθμό βημάτων ανά επεισόδιο επιβάλουμε τον τερματισμό του μετά από κάποιο χρονικό διάστημα. Όσον αφορά τις τιμές των επεισοδίων, του learning rate, του ε_{decay} καθώς και του max_steps_per_episode αυτές υπολογίστηκαν πειραματικά δοκιμάζοντας διάφορους συνδυασμούς έως ότου ο αλγόριθμος καταφέρει να βρει την βέλτιστη πολιτική ή κάτι το οποίο επιφέρει συναθροίστηκα κέρδη που πλησιάζουν στο optimal policy.

Το **δεύτερο σενάριο** περιγράφεται από το εξής περιβάλλον:

- Μετοχές: N = 2
- Transaction Fee: c = 0.05
- Discount Factor: γ = 0.9
- Rewards στο High State: r_H = [0.08, 0.04]
- Rewards στο Low State: r_L = [-0.01, 0.02]
- Πιθανότητες Μετάβασης:
    - Από High σε High State: P_{HH} = [0.9, 0.85]
    - Από High σε Low State: P_{HL} = [0.1, 0.15]
    - Από Low σε High State: P_{LH} = [0.1, 0.2]
    - Από Low σε Low State: P_{LL} = [0.9, 0.8]

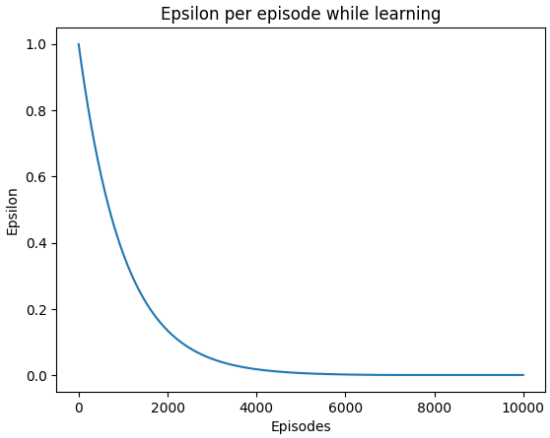
Σε αυτή την περίπτωση είναι εμφανές ότι πλέον δεν αποτελεί βέλτιστη πολιτική απλά η διατήρηση της αρχικής μετοχής καθώς το γ = 0.9 και άρα ενδιαφερόμαστε για μακροχρόνια κέρδη. Επομένως σε αυτό το περιβάλλον αναμένεται κάποιες φορές να διατηρούμε την ίδια μετοχή και κάποιες φορές να είναι προτιμότερη η επιλογή του switch.

Για τον αλγόριθμο Tabular Q Learning οι παράμετροι που τέθηκαν για να επιφέρουν την βέλτιστη πολιτική είναι:

- Episodes: 10000
- Learning Rate: α = 0.1 (δεν χρειάστηκε μείωση με την πάροδο του χρόνου)
- Αρχικό ε για την ε-Greedy επιλογή action: ε = 1.0
- Decay Factor του ε: ε_{decay} = 0.001
- Μέγιστος αριθμός βημάτων ανά επεισόδιο: max_steps_per_episode = 100

Ο τρόπος χειρισμού των επεισοδίων είναι ίδιος με το προηγούμενο σενάριο. Οι παράμετροι του αλγορίθμου και σε αυτή την περίπτωση βρέθηκαν μετά από πειραματικές δοκιμές για διάφορους συνδυασμούς έως ότου βρεθεί η βέλτιστη πολιτική ή κάτι το οποίο επιφέρει συναθροίστηκα κέρδη που πλησιάζουν στο optimal policy.

Και στα δύο σενάρια η παράμετρος ε μειώνεται εκθετικά μετά από κάθε επεισόδιο και συγκεκριμένα όπως η συνάρτηση $f(x) = \varepsilon * e^{-\varepsilon_{decay} * x}, x \in (0, total\_episodes]$.



Αποτελέσματα Tabular Q Learning και Σύγκριση με Ground Truth

Για την εξακρίβωση της ορθότητας των αποτελεσμάτων θα χρησιμοποιήσουμε ως ground truth τα αποτελέσματα του Policy Iteration αλγορίθμου, που αναπτύχθηκε στο πλαίσιο του assignment 3. Ακολουθούν τα policies που προκύπτουν από τον αλγόριθμο του Tabular Q Learning και η σύγκριση με το Policy Iteration.

Σενάριο 1

```
Optimal policy:

State:['1', '1L', '2L']  , action:1
State:['1', '1L', '2H']  , action:1
State:['1', '1H', '2L']  , action:1
State:['1', '1H', '2H']  , action:1
State:['2', '1L', '2L']  , action:2
State:['2', '1L', '2H']  , action:2
State:['2', '1H', '2L']  , action:2
State:['2', '1H', '2H']  , action:2
```

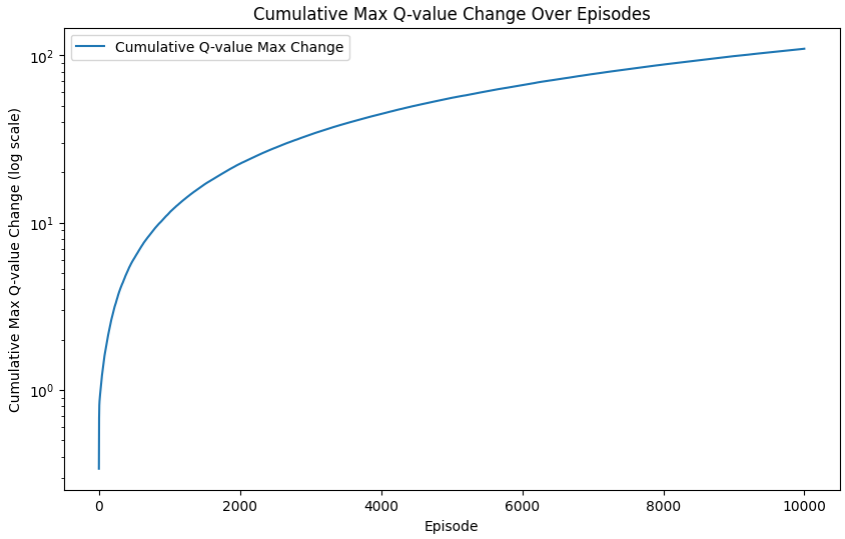
Εικόνα 1 Ground Truth from Policy Iteration

```
Testing the learned policy:

Current State: [1, '1L', '2L'], Action: 1
Current State: [1, '1L', '2H'], Action: 1
Current State: [1, '1H', '2L'], Action: 1
Current State: [1, '1H', '2H'], Action: 1
Current State: [2, '1L', '2L'], Action: 2
Current State: [2, '1L', '2H'], Action: 2
Current State: [2, '1H', '2L'], Action: 2
Current State: [2, '1H', '2H'], Action: 2
```

Εικόνα 2 Learned Policy from Tabular Q Learning

Το χρονικό διάστημα που χρειάστηκε για να συγκλίνει ο αλγόριθμος ήταν περίπου 1 λεπτό. Όπως φαίνεται οι δύο πολιτικές έχουν 100% ταύτιση και ακολουθούν την λογική του να διατηρούν την ίδια μετοχή και να μην αλλάζουν σε άλλη. Επομένως σε αυτή την περίπτωση ο αλγόριθμος έχει επιτύχει την εύρεση της βέλτιστης πολιτικής. Ειδικότερα για τον αλγόριθμο Q Learning έχουμε και το εξής γράφημα:



Εδώ φαίνεται η συναθροιστική μέγιστη διαφορά μεταξύ των Q values ανά επεισόδιο, δηλαδή η μέγιστη διαφορά μεταξύ των τιμών των Q(s,a) μεταξύ δυο επεισοδίων. Όπως φαίνεται η γραφική παράσταση είναι sublinear και φτάνει σε ένα πλατό κοντά στα 6000 επεισόδια πράγμα που φανερώνει ότι οι τιμές των Q(s,a) με την πάροδο του χρόνου αλλάζουν ελάχιστα και άρα έχουμε οδηγηθεί σε σύγκλιση.

Σενάριο 2

```
Optimal policy:

State:['1', '1L', '2L']  , action:2
State:['1', '1L', '2H']  , action:2
State:['1', '1H', '2L']  , action:1
State:['1', '1H', '2H']  , action:1
State:['2', '1L', '2L']  , action:2
State:['2', '1L', '2H']  , action:2
State:['2', '1H', '2L']  , action:1
State:['2', '1H', '2H']  , action:1
```

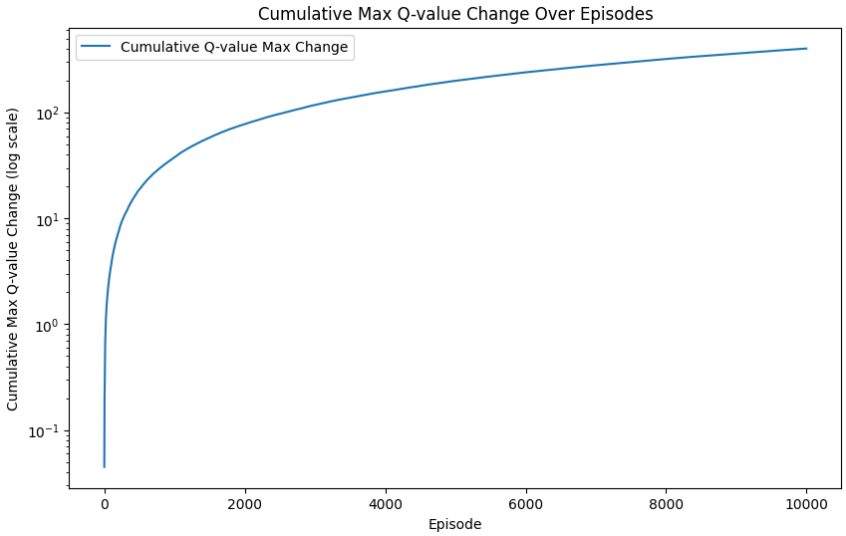
Εικόνα 3 Ground Truth from Policy Iteration

```
Testing the learned policy:

Current State: [1, '1L', '2L'], Action: 2
Current State: [1, '1L', '2H'], Action: 2
Current State: [1, '1H', '2L'], Action: 1
Current State: [1, '1H', '2H'], Action: 1
Current State: [2, '1L', '2L'], Action: 2
Current State: [2, '1L', '2H'], Action: 2
Current State: [2, '1H', '2L'], Action: 1
Current State: [2, '1H', '2H'], Action: 1
```

Εικόνα 4 Learned Policy from Tabular Q Learning

Το χρονικό διάστημα που χρειάστηκε για να συγκλίνει ο αλγόριθμος ήταν περίπου 1 λεπτό. Όπως φαίνεται και σε αυτό το σενάριο ο αλγόριθμος Tabular Q Learning καταφέρνει να πετύχει 100% ταύτιση με την βέλτιστη πολιτική που προκύπτει από τον αλγόριθμο Policy Iteration και άρα έχει επιτύχει την εύρεση της βέλτιστης πολιτικής. Επιπλέον :



Εδώ και πάλι βλέποντας την μορφή της συναθροιστικής μέγιστης διαφοράς μεταξύ των Q values ανά επεισόδιο φαίνεται ότι μετά το επεισόδιο 6000 φτάνουμε σε ένα πλατό και άρα οι αλλαγές στις τιμές των Q(s,a) γίνονται όλο και μικρότερες, οπότε φτάνουμε σε σύγκλιση. Σε αυτό το σενάριο βλέπουμε ότι η διαφορά των Q values ανά επεισόδιο αθροιστικά είναι μεγαλύτερες από ότι στο πρώτο σενάριο. Αυτό οφείλεται στο γεγονός ότι το σενάριο είναι πιο πολύπλοκο σε σχέση με το πρώτο και οι τιμές των Q values είναι σε αυτή την περίπτωση όπου το γ = 0.9 από ότι πριν που το γ = 0 .

Task 2

Το δεύτερο task αφορά την εφαρμογή του Tabular Q Learning αλγορίθμου που αναπτύξαμε πριν σε ένα πιο πολύπλοκο περιβάλλον. Εδώ το σενάριο που χρησιμοποιήθηκε είναι το ίδιο με το Question 3 του Assignment 3.

Συγκεκριμένα το σενάριο περιγράφεται από το εξής περιβάλλον:

- Μετοχές: N = 8
- Rewards: κάθε r_i^H, r_i^L επιλέγεται uniformly στο διάστημα [-0.02, +0.1]
- Πιθανότητες Μετάβασης: οι πιθανότητες μετάβασης P_{HL}^i, P_{LH}^i είναι 0.1 για τις μισές μετοχές και 0.5 για τις άλλες μισές
- Discount Factor: γ = 0.95
- Transaction Fee: c = 0.05

Για τον αλγόριθμο Tabular Q Learning οι παράμετροι που τέθηκαν για να επιφέρουν την βέλτιστη πολιτική είναι:

- Episodes: 10000
- Learning Rate: α = 0.1 (δεν χρειάστηκε μείωση με την πάροδο του χρόνου)
- Αρχικό ε για την ε-Greedy επιλογή action: ε = 1.0
- Decay Factor του ε: ε_{decay} = 0.001
- Μέγιστος αριθμός βημάτων ανά επεισόδιο: max_steps_per_episode = 100

Ο τρόπος χειρισμού των επεισοδίων είναι ίδιος με τα 2 προηγούμενα σενάρια. Οι παράμετροι του αλγορίθμου και σε αυτή την περίπτωση βρέθηκαν μετά από πειραματικές δοκιμές για διάφορους συνδυασμούς έως ότου βρεθεί η βέλτιστη πολιτική ή κάτι το οποίο επιφέρει συναθροίστηκα κέρδη που πλησιάζουν στο optimal policy.

Η παράμετρος ε μειώνεται εκθετικά μετά από κάθε επεισόδιο και συγκεκριμένα όπως η συνάρτηση $f(x) = \frac{\varepsilon}{1+\varepsilon_{decay}*x}, x \in (0, total\_episodes]$. Η ελάχιστη τιμή που μπορεί να πάρει το ε έχει τεθεί σε 0.0001. Η συγκεκριμένη μείωση του ε προέκυψε πειραματικά δοκιμάζοντας εκτός της υπάρχουσας γραμμική και εκθετική μείωση με την υπάρχουσα να επιφέρει τα καλύτερα αποτελέσματα.

Επειδή η κάθε εκτέλεση του αλγορίθμου δημιουργεί περιβάλλοντα με διάφορες τιμές reward και πιθανοτήτων μετάβασης, για να μπορέσουμε να έχουμε ένα σταθερό μέτρο σύγκρισης και να μπορέσουμε να μελετήσουμε την ορθότητα της πολικής που προκύπτει από τον Q Learning αλγόριθμο θα χρησιμοποιήσουμε το ενδεικτικό περιβάλλον:

- Rewards:
  - r_H= [0.09191125, 0.00712202, 0.08180192, 0.0991868, 0.07771591, 0.09109335, 0.08568778, 0.07824772]
  - r_L = [0.03969432, -0.00729206, 0.05855026, 0.00179933, 0.07403611, 0.04170629, 0.05884049, 0.04389849]

- Πιθανότητες Μετάβασης:
  - P_{HH} = [0.9, 0.9, 0.9, 0.9, 0.5, 0.5, 0.5, 0.5]
  - P_{HL} = [0.1, 0.1, 0.1, 0.1, 0.5, 0.5, 0.5, 0.5]
  - P_{LL} = [0.5, 0.5, 0.5, 0.5, 0.9, 0.9, 0.9, 0.9]
  - P_{LH} = [0.5, 0.5, 0.5, 0.5, 0.1, 0.1, 0.1, 0.1]

Αποτελέσματα Tabular Q Learning και Σύγκριση με Ground Truth

Και εδώ για τον έλεγχο ορθότητας θα θεωρήσουμε ως Ground Truth τα αποτελέσματα του Policy Iteration από το Assignment 3 για το δεδομένο περιβάλλον.

Αρχικά για να δείξουμε ότι το περιβάλλον δεν είναι Trivial παρακάτω φαίνεται ο αριθμός των φορών που έχει επιλεχθεί μία μετοχή με βάση την βέλτιστη πολιτική:

```
Stocks Chosen: [800.   0. 160. 896.  64.  32.  64.  32.]
```

Με βάση τον πίνακα επιλογών τα ποσοστά επιλογών μετοχών είναι:

Stock 1	Stock 2	Stock 3	Stock 4	Stock 5	Stock 6	Stock 7	Stock 8
39.06%	0%	7.81%	43.75%	3.13%	1.56%	3.13%	1.56%

Επειδή το πλήθος των καταστάσεων είναι αρκετά μεγάλο παρακάτω παρατίθενται μερικά από τα αποτελέσματα:

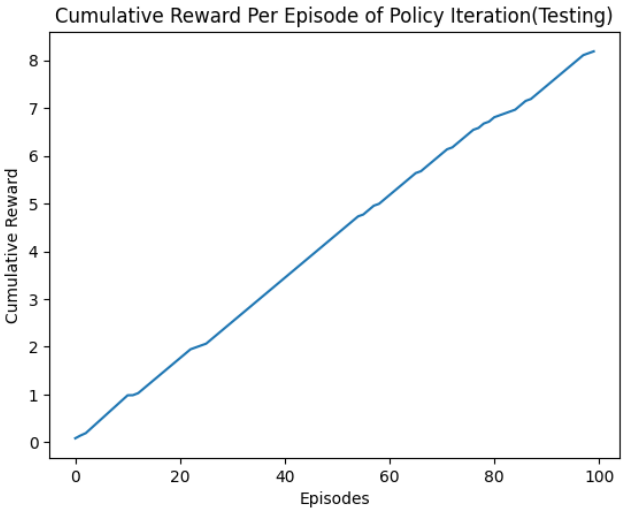
```
State:['2', '1L', '2H', '3H', '4L', '5H', '6L', '7H', '8L']  , action:1
State:['2', '1L', '2H', '3H', '4L', '5H', '6L', '7H', '8H']  , action:1
State:['2', '1L', '2H', '3H', '4L', '5H', '6H', '7L', '8L']  , action:1
State:['2', '1L', '2H', '3H', '4L', '5H', '6H', '7L', '8H']  , action:1
State:['2', '1L', '2H', '3H', '4L', '5H', '6H', '7H', '8L']  , action:1
State:['2', '1L', '2H', '3H', '4L', '5H', '6H', '7H', '8H']  , action:1
State:['2', '1L', '2H', '3H', '4H', '5L', '6L', '7L', '8L']  , action:4
State:['2', '1L', '2H', '3H', '4H', '5L', '6L', '7L', '8H']  , action:4
State:['2', '1L', '2H', '3H', '4H', '5L', '6L', '7H', '8L']  , action:4
State:['2', '1L', '2H', '3H', '4H', '5L', '6L', '7H', '8H']  , action:4
State:['2', '1L', '2H', '3H', '4H', '5L', '6H', '7L', '8L']  , action:4
State:['2', '1L', '2H', '3H', '4H', '5L', '6H', '7L', '8H']  , action:4
State:['2', '1L', '2H', '3H', '4H', '5L', '6H', '7H', '8L']  , action:4
State:['2', '1L', '2H', '3H', '4H', '5L', '6H', '7H', '8H']  , action:4
State:['2', '1L', '2H', '3H', '4H', '5H', '6L', '7L', '8L']  , action:4
State:['2', '1L', '2H', '3H', '4H', '5H', '6L', '7L', '8H']  , action:4
State:['2', '1L', '2H', '3H', '4H', '5H', '6L', '7H', '8L']  , action:4
State:['2', '1L', '2H', '3H', '4H', '5H', '6L', '7H', '8H']  , action:4
State:['2', '1L', '2H', '3H', '4H', '5H', '6H', '7L', '8L']  , action:4
State:['2', '1L', '2H', '3H', '4H', '5H', '6H', '7L', '8H']  , action:4
State:['2', '1L', '2H', '3H', '4H', '5H', '6H', '7H', '8L']  , action:4
State:['2', '1L', '2H', '3H', '4H', '5H', '6H', '7H', '8H']  , action:4
State:['2', '1H', '2L', '3L', '4L', '5L', '6L', '7L', '8L']  , action:1
```

Εικόνα 5 Ground Truth Policy Sample from Policy Iteration

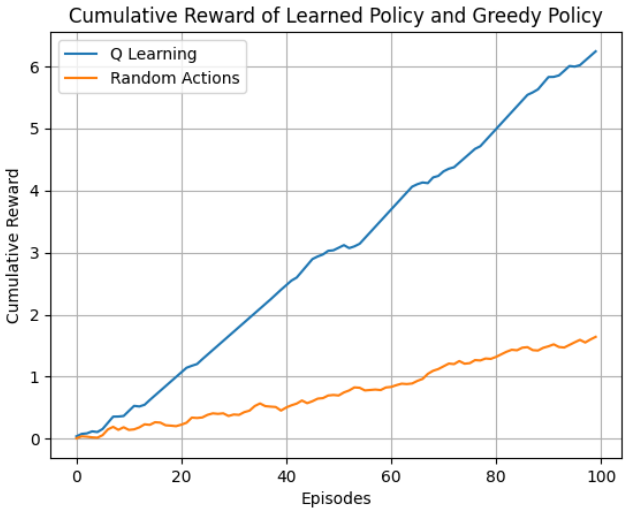
```
Current State: [2, '1L', '2H', '3H', '4L', '5H', '6L', '7H', '8L'], Action: 1
Current State: [2, '1L', '2H', '3H', '4L', '5H', '6L', '7H', '8H'], Action: 1
Current State: [2, '1L', '2H', '3H', '4L', '5H', '6H', '7L', '8L'], Action: 1
Current State: [2, '1L', '2H', '3H', '4L', '5H', '6H', '7L', '8H'], Action: 1
Current State: [2, '1L', '2H', '3H', '4L', '5H', '6H', '7H', '8L'], Action: 1
Current State: [2, '1L', '2H', '3H', '4L', '5H', '6H', '7H', '8H'], Action: 1
Current State: [2, '1L', '2H', '3H', '4H', '5L', '6L', '7L', '8L'], Action: 7
Current State: [2, '1L', '2H', '3H', '4H', '5L', '6L', '7L', '8H'], Action: 4
Current State: [2, '1L', '2H', '3H', '4H', '5L', '6L', '7H', '8L'], Action: 4
Current State: [2, '1L', '2H', '3H', '4H', '5L', '6L', '7H', '8H'], Action: 1
Current State: [2, '1L', '2H', '3H', '4H', '5L', '6H', '7L', '8L'], Action: 3
Current State: [2, '1L', '2H', '3H', '4H', '5L', '6H', '7L', '8H'], Action: 1
Current State: [2, '1L', '2H', '3H', '4H', '5L', '6H', '7H', '8L'], Action: 2
Current State: [2, '1L', '2H', '3H', '4H', '5L', '6H', '7H', '8H'], Action: 1
Current State: [2, '1L', '2H', '3H', '4H', '5H', '6L', '7L', '8L'], Action: 3
Current State: [2, '1L', '2H', '3H', '4H', '5H', '6L', '7L', '8H'], Action: 1
Current State: [2, '1L', '2H', '3H', '4H', '5H', '6L', '7H', '8L'], Action: 1
Current State: [2, '1L', '2H', '3H', '4H', '5H', '6L', '7H', '8H'], Action: 8
Current State: [2, '1L', '2H', '3H', '4H', '5H', '6H', '7L', '8L'], Action: 1
Current State: [2, '1L', '2H', '3H', '4H', '5H', '6H', '7L', '8H'], Action: 1
Current State: [2, '1L', '2H', '3H', '4H', '5H', '6H', '7H', '8L'], Action: 1
Current State: [2, '1L', '2H', '3H', '4H', '5H', '6H', '7H', '8H'], Action: 1
Current State: [2, '1H', '2L', '3L', '4L', '5L', '6L', '7L', '8L'], Action: 3
```

Εικόνα 6 Learned Policy from Tabular Q Learning

Όπως είναι εμφανές οι δύο πολιτικές ακόμη και σε αυτό το μικρό sample φέρουν κάποιες διαφορές. Εδώ θα πρέπει να συγκρίνουμε τα αθροιστικά rewards ώστε να δούμε εάν ο αλγόριθμος Tabular Q Learning καταφέρνει να βρει αν όχι την βέλτιστη πολιτική τουλάχιστον μια εξίσου καλή. Για να κάνουμε αυτό τον έλεγχο τρέξαμε για 100 steps, ξεκινώντας από μία τυχαία κατάσταση και ακολουθώντας στην μία περίπτωση την πολιτική του policy iteration και στην άλλη αυτή του Tabular Q Learning και συγκρίναμε τα συνολικά κέρδη τους. Επιπλέον στο γράφημα των reward του Q Learning φαίνεται και η περίπτωση που ακολουθούμε μια πολιτική όπου κάνουμε τυχαίες επιλογές stock σε κάθε γύρο. Τα αποτελέσματα ήταν τα εξής:



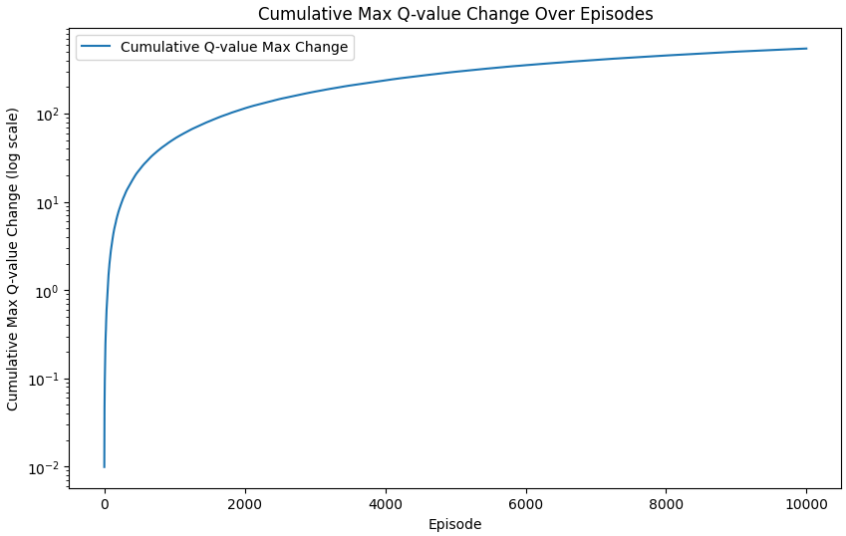
Εικόνα 7 Συνολικά κέρδη policy από το Policy Iteration



Εικόνα 8 Συνολικά κέρδη policy από Tabular Q Learning

Όπως φαίνεται από τα γραφήματα είναι εμφανές ότι η περίπτωση που διαλέγουμε random actions είναι κατά πολύ suboptimal, πράγμα που επιβεβαιώνει ότι το πρόβλημα για το δεδομένο περιβάλλον δεν είναι Trivial. Το policy του αλγορίθμου Tabular Q Learning παρόλο που κάνει διαφορετικές επιλογές από ότι το optimal policy, που προκύπτει από το Policy Iteration, εν τέλει καταφέρνει να επιτύχει ένα καλό αριθμό κερδών λίγο πάνω από 6€ μόλις 2€ κάτω από αυτό που πετυχαίνει το optimal policy, δηλαδή λίγο πάνω από 8€. Για αυτήν τη διαφορά πιθανόν να οφείλεται και το γεγονός , ότι τα yields των μετοχών ανά ημέρα είναι random και διαφορετικά μεταξύ τους στα δύο σενάρια.

Στο παρακάτω γράφημα φαίνεται η μέγιστη διαφορά στα Q values ανά επεισόδιο:



Σε αυτό το γράφημα φαίνεται ότι ο αλγόριθμος έχει οδηγηθεί σε σύγκλιση, καθώς μετά το επεισόδιο 7000 η μέγιστη διαφορά μεταξύ των τιμών των Q values είναι πολύ μικρή. Αυτό και επιπλέον το γεγονός ότι η γραφική είναι sublinear δείχνει ότι ο αλγόριθμος επιτυγχάνει και μαθαίνει μια πολιτική η οποία ναι μεν δεν είναι η βέλτιστη αλλά είναι μια αρκετά καλή δεδομένου του αριθμού κερδών που επιτυγχάνει. Ο συνολικός χρόνος που χρειάστηκε για την σύγκλιση του Q-Learning αλγορίθμου ήταν 24 λεπτά , ενώ για το PI περίπου 35 λεπτά.

Task 3 – DQN

Σε αυτό το task προσπαθούμε να επιλύσουμε τις αδυναμίες του tabular q learning , δουλεύοντας με νευρωνικά δίκτυα και deep RL methods.Συγκεκριμένα υλοποιήσαμε ένα Double DQN αλγόριθμο καθώς χρησιμοποιήσαμε δύο ξεχωριστά Neural Networks, με το πρώτο να αντιστοιχεί στο policy και το δεύτερο στο target.

- **Αρχιτεκτονική:** Και τα δύο NN αποτελούνται από 3 layers (ένα input, ένα hidden και ένα output), οι διαστάσεις των οποίων είναι:
  - Το πρώτο (N+1)x64 . Η είσοδος του πρώτου layer αντιστοιχεί στο state το οποίο είναι ένα Tensor διάστασης 1xN+1 όπου το πρώτο στοιχείο είναι η μετοχή στην οποία βρίσκομαι και τα υπόλοιπα στοιχεία αντιστοιχούν στις τιμές όλων των μετοχών εκείνη την στιγμή.
  - Το δεύτερο αποτελεί ένα hidden layer διάστασης 64x64.
  - Το τρίτο , το οποίο αποτελεί την έξοδο του νευρωνικού , είναι διάστασης 64xN , και η έξοδος είναι 1xN και αντιστοιχεί σε όλα τα Q values για το δεδομένο state και action την μετάβαση σε stock με δείκτη 0 έως N-1.

- **Hyperparameters:**
  - Learning rate α = 0.01
  - ε = 1.0
  - $\varepsilon_{decay} = 0.95$
  - episodes = 100
  - max_steps_per_episode = 150,
  - replay_buffer_size = 1000
  - batch_size = 64

  Οι παράμετροι α, ε_{decay}, episodes, max_steps_per_episode υπολογίστηκαν πειραματικά τρέχοντας τον αλγόριθμο έως ότου να βρεθεί το σωστό policy και να ικανοποιείται το Trade Off μεταξύ χρόνου και convergence.

  Για την εύρεση της βέλτιστης πολιτικής ο αλγόριθμος τρέχει 100 επεισόδια και για κάθε επεισόδιο τρέχει έναν σχετικά μεγαλύτερο αριθμό βημάτων που όταν συμπληρωθούν τερματίζει το κάθε επεισόδιο και γίνεται reset το περιβάλλον.

  Επίσης κάθε 10 επεισόδια γίνεται συγχρονισμός των δύο νευρωνικών δικτύων του αλγορίθμου.

- Το **περιβάλλον** παραμένει ίδιο με αυτό του Task 2 .

Αποτελέσματα DQN και Σύγκριση με Ground Truth

1. Έλεγχος ορθότητας για μικρό περιβάλλον

Αρχικά για να μπορέσουμε να είμαστε βέβαιοι ότι το DQN λειτουργεί και μπορεί να βρει το optimal policy ή να το πλησιάσει αρκετά δοκιμάσαμε το περιβάλλον του question 1 από το assignment 3. Για την δοκιμή τρέχουμε για την περίπτωση που το αρχικό stock που επενδύσαμε είναι το 0 ,δέκα step στο περιβάλλον και το ίδιο για την περίπτωση που το αρχικό stock είναι το 1. Αυτό που αναμένουμε είναι να παραμένει στο ίδιο stock που ξεκίνησε σε κάθε περίπτωση.

```
Testing Policy:
Day 1: Stock 0, Current Stock Rewards: [1 1], Action: 0, Next Stock Rewards: [1 1], Reward: 0.08
Day 2: Stock 0, Current Stock Rewards: [1 1], Action: 0, Next Stock Rewards: [1 1], Reward: 0.08
Day 3: Stock 0, Current Stock Rewards: [1 1], Action: 0, Next Stock Rewards: [1 1], Reward: 0.08
Day 4: Stock 0, Current Stock Rewards: [0 1], Action: 0, Next Stock Rewards: [0 1], Reward: 0.08
Day 5: Stock 0, Current Stock Rewards: [0 1], Action: 0, Next Stock Rewards: [0 1], Reward: 0.01
Day 6: Stock 0, Current Stock Rewards: [0 1], Action: 0, Next Stock Rewards: [0 1], Reward: 0.01
Day 7: Stock 0, Current Stock Rewards: [0 1], Action: 0, Next Stock Rewards: [0 1], Reward: 0.01
Day 8: Stock 0, Current Stock Rewards: [0 1], Action: 0, Next Stock Rewards: [0 1], Reward: 0.01
Day 9: Stock 0, Current Stock Rewards: [0 1], Action: 0, Next Stock Rewards: [0 1], Reward: 0.01
Day 10: Stock 0, Current Stock Rewards: [0 1], Action: 0, Next Stock Rewards: [0 1], Reward: 0.01
Day 1: Stock 1, Current Stock Rewards: [1 0], Action: 1, Next Stock Rewards: [1 0], Reward: -0.02
Day 2: Stock 1, Current Stock Rewards: [1 0], Action: 1, Next Stock Rewards: [1 0], Reward: -0.02
Day 3: Stock 1, Current Stock Rewards: [1 0], Action: 1, Next Stock Rewards: [1 0], Reward: -0.02
Day 4: Stock 1, Current Stock Rewards: [0 0], Action: 1, Next Stock Rewards: [0 0], Reward: -0.02
Day 5: Stock 1, Current Stock Rewards: [0 0], Action: 1, Next Stock Rewards: [0 0], Reward: -0.02
Day 6: Stock 1, Current Stock Rewards: [0 0], Action: 1, Next Stock Rewards: [0 0], Reward: -0.02
Day 7: Stock 1, Current Stock Rewards: [0 0], Action: 1, Next Stock Rewards: [0 0], Reward: -0.02
Day 8: Stock 1, Current Stock Rewards: [0 0], Action: 1, Next Stock Rewards: [0 0], Reward: -0.02
Day 9: Stock 1, Current Stock Rewards: [0 0], Action: 1, Next Stock Rewards: [0 0], Reward: -0.02
Day 10: Stock 1, Current Stock Rewards: [0 0], Action: 1, Next Stock Rewards: [0 0], Reward: -0.02
```

Εικόνα 9 Αποτέλεσμα 10 step ξεκινώντας αρχικά από stock 0 , και έπειτα από stock 1.

Όπως φαίνεται και από τα αποτελέσματα, το policy που ακολουθείται είναι το ίδιο με το assignment 3 και άρα ξέρουμε ότι το DQN δουλεύει σωστά, τουλάχιστον για τα πιο μικρά σενάρια.

2. Έλεγχος ορθότητας για μεγάλο περιβάλλον.

Για τον έλεγχο ορθότητας του policy σε μεγαλύτερο περιβάλλον , χρησιμοποιήσαμε το ενδεικτικό περιβάλλον που είχαμε χρησιμοποιήσει για σύγκριση και στο Task 2, για N = 8.Για να συγκρίνουμε το policy του DQN με το ground truth policy που προκύπτει από το PI , μετά το training , κάνουμε exploit για 100 ημέρες (100 steps στο περιβάλλον) ξεκινώντας από ένα random state , και με random yields μετοχών ανά ημέρα. Ενδεικτικά , παρατίθενται μερικά από τα action (με αρίθμηση stocks από 0 έως N -1) που διάλεξε το policy σε μερικές ημέρες:

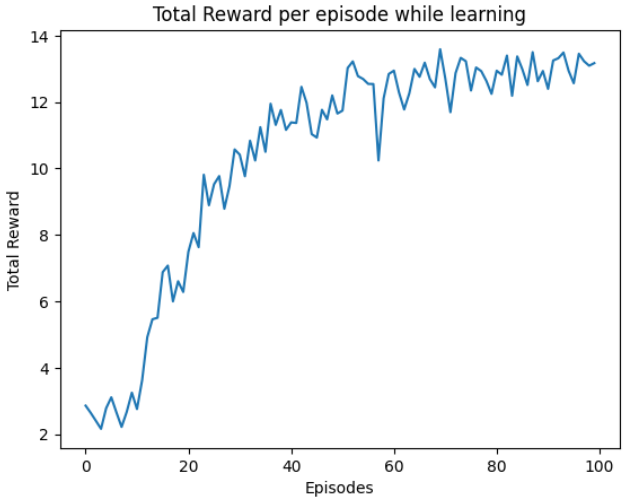
```
Day 17: Stock 3, Current Stock Rewards: [1 1 0 1 0 0 0 0], Action: 3, Next Stock Rewards: [1 1 0 1 0 0 0 0], Reward: 0.0991868
Day 18: Stock 3, Current Stock Rewards: [1 1 1 1 0 0 0 0], Action: 3, Next Stock Rewards: [1 1 1 1 0 0 0 0], Reward: 0.0991868
Day 19: Stock 3, Current Stock Rewards: [1 0 1 1 0 0 0 0], Action: 3, Next Stock Rewards: [1 0 1 1 0 0 0 0], Reward: 0.0991868
Day 20: Stock 3, Current Stock Rewards: [1 1 0 1 0 0 1 0], Action: 3, Next Stock Rewards: [1 1 0 1 0 0 1 0], Reward: 0.0991868
Day 21: Stock 3, Current Stock Rewards: [1 1 0 1 0 0 1 0], Action: 3, Next Stock Rewards: [1 1 0 1 0 0 1 0], Reward: 0.0991868
Day 22: Stock 3, Current Stock Rewards: [0 0 1 1 0 0 0 0], Action: 3, Next Stock Rewards: [0 0 1 1 0 0 0 0], Reward: 0.0991868
Day 23: Stock 3, Current Stock Rewards: [1 0 1 1 0 0 0 0], Action: 3, Next Stock Rewards: [1 0 1 1 0 0 0 0], Reward: 0.0991868
Day 24: Stock 3, Current Stock Rewards: [1 1 1 1 0 0 0 0], Action: 3, Next Stock Rewards: [1 1 1 1 0 0 0 0], Reward: 0.0991868
Day 25: Stock 3, Current Stock Rewards: [1 0 1 0 0 0 0 1], Action: 3, Next Stock Rewards: [1 0 1 0 0 0 0 1], Reward: 0.0991868
Day 26: Stock 3, Current Stock Rewards: [1 0 1 0 0 1 0 0], Action: 2, Next Stock Rewards: [1 0 1 0 0 1 0 0], Reward: 0.03180192
Day 27: Stock 2, Current Stock Rewards: [1 1 1 0 0 0 1 0], Action: 2, Next Stock Rewards: [1 1 1 0 0 0 1 0], Reward: 0.08180192
Day 28: Stock 2, Current Stock Rewards: [1 1 1 1 1 1 1 0], Action: 2, Next Stock Rewards: [1 1 1 1 1 1 1 0], Reward: 0.08180192
Day 29: Stock 2, Current Stock Rewards: [1 1 1 1 0 0 0 0], Action: 3, Next Stock Rewards: [1 1 1 1 0 0 0 0], Reward: 0.0491868
Day 30: Stock 3, Current Stock Rewards: [1 1 0 1 1 0 0 0], Action: 3, Next Stock Rewards: [1 1 0 1 1 0 0 0], Reward: 0.0991868
Day 31: Stock 3, Current Stock Rewards: [1 1 0 1 0 1 0 0], Action: 3, Next Stock Rewards: [1 1 0 1 0 1 0 0], Reward: 0.0991868
Day 32: Stock 3, Current Stock Rewards: [1 1 0 1 0 1 0 0], Action: 3, Next Stock Rewards: [1 1 0 1 0 1 0 0], Reward: 0.0991868
Day 33: Stock 3, Current Stock Rewards: [1 1 1 1 0 1 0 0], Action: 3, Next Stock Rewards: [1 1 1 1 0 1 0 0], Reward: 0.0991868
Day 34: Stock 3, Current Stock Rewards: [1 1 1 1 0 0 0 0], Action: 3, Next Stock Rewards: [1 1 1 1 0 0 0 0], Reward: 0.0991868
```

Εικόνα 10 Μερικά action του DQN policy

Policy Ground Truth από PI (αρίθμηση των action από 1 έως N) για τις παραπάνω ενδεικτικές μεταβάσεις:

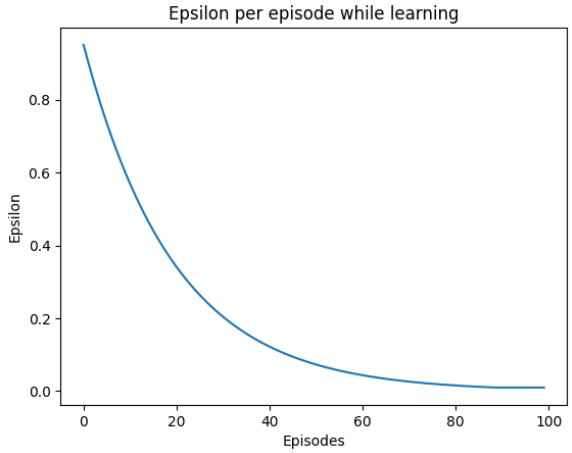
```
State:['4', '1H', '2H', '3L', '4H', '5L', '6L', '7L', '8L']  , action:4
State:['4', '1H', '2H', '3H', '4H', '5L', '6L', '7L', '8L']  , action:4
State:['4', '1H', '2L', '3H', '4H', '5L', '6H', '7L', '8L']  , action:4
State:['4', '1H', '2H', '3L', '4H', '5L', '6L', '7H', '8L']  , action:4
State:['4', '1L', '2L', '3H', '4H', '5L', '6L', '7L', '8L']  , action:4
State:['4', '1H', '2L', '3H', '4L', '5L', '6L', '7L', '8H']  , action:1
State:['4', '1H', '2L', '3H', '4L', '5L', '6H', '7L', '8L']  , action:1
State:['3', '1H', '2H', '3H', '4L', '5L', '6L', '7H', '8L']  , action:3
State:['3', '1H', '2H', '3H', '4H', '5H', '6H', '7H', '8L']  , action:3
State:['3', '1H', '2H', '3H', '4H', '5L', '6L', '7L', '8L']  , action:3
State:['4', '1H', '2H', '3L', '4H', '5H', '6L', '7L', '8L']  , action:4
State:['4', '1H', '2H', '3L', '4H', '5L', '6H', '7L', '8L']  , action:4
State:['4', '1H', '2H', '3H', '4H', '5L', '6H', '7L', '8L']  , action:4
```

Όπως είναι εμφανές οι δύο πολιτικές ακόμη και σε αυτό το μικρό sample φέρουν κάποιες διαφορές .Όμως, όπως θα δούμε στη σύγκριση παρακάτω , τα overall κέρδη είναι παρεμφερή. Παρακάτω φαίνεται το συνολικό κέρδος ανά επεισόδιο (150 steps/επεισόδιο) για το ενδεικτικό περιβάλλον κατά το training:



Το γεγονός ότι τα κέρδη αυξάνονται ανά επεισόδιο μας δείχνει ότι ο DQN agent μαθαίνει και κάνει πιο σωστές επιλογές ανά επεισόδιο , άρα συνολικά ότι το DQN δουλεύει. Επιπλέον μπορούμε να παρατηρήσουμε ότι μετά περίπου από 60 επεισόδια το κέρδος φτάνει σε ένα

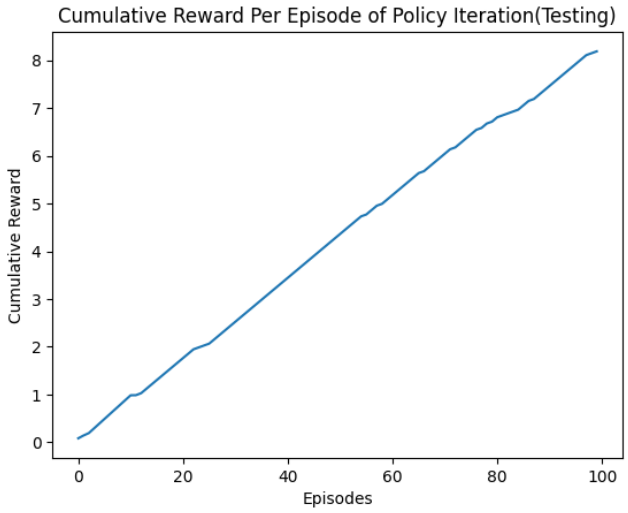
πλατό και άρα έχουμε πρακτικά οδηγηθεί σε σύγκλιση. Παρακάτω παρατίθεται το διάγραμμα τιμών του ε ανά επεισόδιο μάθησης.

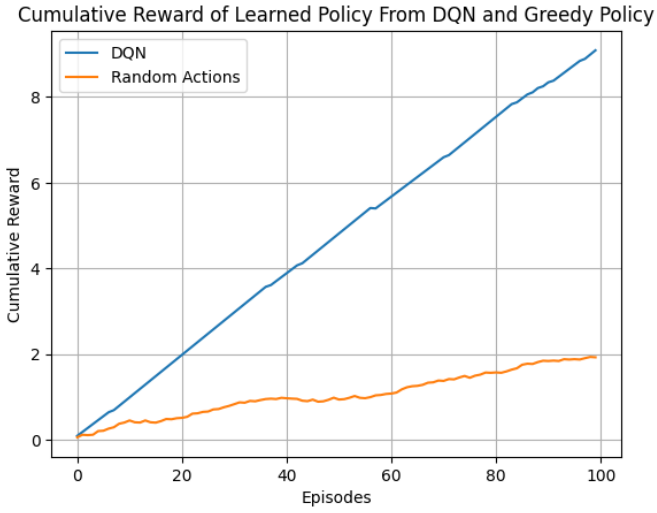


Η τιμή του ε μειώνεται εκθετικά όπως η $f(x) = \varepsilon * (\varepsilon_{decay})^x, x \in (0, total\_episodes]$, και μετά περίπου από 40 επεισόδια τελειώνει το exploration phase , έως ότου φτάσει στην ελάχιστη επιτρεπτή δυνατή τιμή 0.001.

Σύγκριση κέρδους DQN - PI

Εδώ θα πρέπει να συγκρίνουμε τα αθροιστικά rewards ώστε να δούμε εάν ο αλγόριθμος DQN καταφέρνει να βρει αν όχι την βέλτιστη πολιτική τουλάχιστον μια εξίσου καλή. Για να κάνουμε αυτό τον έλεγχο τρέξαμε για 100 steps, ξεκινώντας από μία τυχαία κατάσταση και ακολουθώντας στην μία περίπτωση την πολιτική του policy iteration και στην άλλη αυτή του DQN και συγκρίναμε τα συνολικά κέρδη τους. Επιπλέον στο γράφημα των reward του DQN φαίνεται και η περίπτωση που ακολουθούμε μια πολιτική όπου κάνουμε τυχαίες επιλογές stock σε κάθε γύρο. Τα αποτελέσματα ήταν τα εξής:





Όπως βλέπουμε , το policy του DQN δεν είναι trivial καθώς διαφέρει πολύ από το dummy random policy. Επιπλέον , συμπεραίνουμε ότι το policy του DQN έρχεται πολύ κοντά με το optimal policy , καθώς τελικά βγάζουν παρεμφερές κέρδος. Ο συνολικός χρόνος που χρειάστηκε το DQN για να βρει το policy ήταν 7 λεπτά, ενώ του αλγορίθμου tabular Q learning 24 λεπτά. Δηλαδή το μοντέλο με το DQN ήταν περίπου 3.5 φορές πιο γρήγορο από το tabular Q learning και σαφώς πιο memory efficient.