# Αυτόνομοι Πράκτορες

Μερσινιάς Μιχαήλ, 2013030057

## ΠΛΗ 513 - Χειμερινό Εξάμηνο 2017-18

## Αναφορά του Project

**Περιγραφή του πρόβληματος:**

Το OpenAI Gym είναι μια μεγάλη σουίτα προβλημάτων (domains) της ομάδας OpenAI για την μελέτη αλγορίθμων ενισχυτικής μάθησης. Το domain το οποίο επέλεξα για το project του μαθήματος ήταν το MountainCar-v0.

Το MountainCar-v0 ορίζεται ως εξής από την βιβλιογραφία του OpenAI Gym:

Goal = “Get an under powered car to the top of a hill (top -> position = 0.5)”.

Observation = {position, velocity}

- Position = [-1.2 , 0.6]
- Velocity = [-0.07 , 0.07]

Action = {Left, Neutral, Right}

- Left = -1
- Neutral = 0
- Right = 1

Reward = “-1 for each step, until the goal position of 0.5 is reached. There is no penalty for climbing the left hill”.

Starting state: “Random position from -0.6 to -0.4. Zero velocity.”

Episode Termination: “The episode ends when you reach 0.5 position, or if 200 iterations are reached”.

Solved requirements: “Get an average reward of -110.0 over 100 consecutive trials”.

Leaderboard:

- jing582: 1119 episodes before solve
- DaveLeongSingapore: 1967 episodes before solve
- joschu: 3962 episodes before solve

Αλλαγές:

Υπήρχαν δύο διαφορές σχετικά με την βιβλιογραφία στο imported MountainCar-v0 environment.

Αντί για Left = 0, Neutral = 1, Right = 2, οι αντίστοιχοι αριθμοί ήταν Left = -1, Neutral = 0, Right = 1. Χρησιμοποιήθηκε, όμως, η εξής καθολική συνάρτηση για την επιλογή action και απλά υπολογίζονταν οι εκάστοτε πιθανότητες κάθε φορά:

“action = numpy.random.choice(env.action_space.n, p=[p_left, p_neutral, p_right])”

Επίσης, σε αντίθεση με την βιβλιογραφία, το reward ήταν -1 ακόμα και για τα steps που γινόντουσαν στον αριστερό λόφο με αρνητικό velocity. Αυτό διορθώθηκε με μια IF η οποία δεν δίνει penalty για την ανάβαση του αριστερού λόφου, όπως ορίζει η βιβλιογραφία.

Τέλος, ως μια επιπλέον δυσκολία, ορίστηκε σταθερό -200 reward στα episodes στα οποία το αμάξι δεν καταφέρνει να ανέβει τον λόφο και να φτάσει το goal. Δηλαδή, κανονικά θα έπαιρνε ένα reward της τάξης του 165 υποθέτοντας ότι 35 περίπου actions έχουν ως αποτέλεσμα observations position<-0.5 και velocity<0. Αλλά στην δική μας περίπτωση θα πάρει -200.

**Προσέγγιση της λύσης:**

Ο αλγόριθμος ενισχυτικής μάθησης που επιλέχτηκε είναι Q-Learning. Ο αλγόριμος αυτός χρησιμοποιεί το παρακάτω update function και λειτουργεί ως εξής.

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t))$$

Πίνακας Q:

Ο πίνακας Q περιέχει τιμές για όλους τους δυνατούς συνδυασμούς των observation μας (position και velocity) οι οποίες αρχικοποιούνται στο μηδέν. Δηλαδή είναι μεγέθους Nx3 όπου N=72*(18-1)+72=1296 states x 3 actions. Το 72 είναι τα velocity states και το 18 είναι τα position states.

Άρα για παράδειγμα, το position state 0 θα καταλαμβάνει τις θέσεις 0..71 του πίνακα Q με την καθεμία από αυτές να αντιστοιχεί σε ένα velocity state. Το position state 1 θα καταλαμβάνει τις θέσεις 72..143 του πίνακα Q κ.ο.κ. Δηλαδή το position m θα καταλαμβάνει τις θέσεις 72*m…72*m+71.

Διακριτικοποίηση:

Ο πίνακας Q χρειάζεται ως όρισμα state και action. Τα actions είναι ήδη διακριτά, αλλά τα state πρέπει να προκύψουν από τα συνεχή φάσματα των δύο observations. Επομένως, δημιουργήθηκε μια συνάρτηση η οποία σπάει ισόποσα το φάσμα του position σε 18 καταστάσεις και το φάσμα του velocity σε 72 καταστάσεις. Γενικά, μετά από δοκιμές παρατηρήθηκε ότι όσο περισσότερες καταστάσεις, τόσο πιο καλά και smooth είναι τα αποτελέσματα. Αλλά ως trade-off, τόσο περισσότερο χρόνο χρειάζονται για να παραχθούν. Αντιθέτως, με λίγες καταστάσεις, η μάθηση είναι γρήγορη αλλά κακή. Τελικά, αποφασίστηκε να υπάρξουν πολλές καταστάσεις για το velocity και πιο λίγες για το position. Διότι όσον αφορά το position, ενδιαφερόμαστε κυρίως για το αν είμαστε αριστερά, δεξιά ή κεντρικά. Αλλά όσον αφορά το velocity, θέλουμε να ξέρουμε την ακριβή του τιμή διότι η παραμικρή της αλλαγή μπορεί να αλλάξει τα δεδομένα.

Επιλογή action:

Αρχικά, λοιπόν, επιλέγουμε ένα τυχαίο action. Αφού αυτό εκτελεστεί, σύμφωνα με το observation και το reward που λαμβάνουμε, κάνουμε update την αντίστοιχη τιμή του πίνακα Q. Από εδώ και πέρα, το action μας θα επιλέγεται σύμφωνα με την μεγαλύτερη τιμή του Q table. Για παράδειγμα, για position=0.2 και velocity=0.01, ο πίνακας Q έχει την τιμή x για action=left, την τιμή y για action=neutral και την τιμή z για action=right. Αν υποθέσουμε ότι x>y και x>z, τότε θα επιλεγεί το action=left με πιθανότητα 1. Εάν x=y=z, θέτουμε πιθανότητα 0.35 για action=right, 0.35 για action=left και 0.3 για action=neutral, δηλαδή ισοπίθανα με μια ελαφρά προτίμηση στην κίνηση παρά στην στασιμότητα. Τις ίδιες ισοπίθανες (σχεδόν) πιθανότητες θέτουμε και με κάποιον βαθμό τυχαιότητας (5% αρχικά και φθίνει με τον χρόνο, συγκεκριμένα 0.01 κάθε 100 επεισόδια) σύμφωνα με τον οποίον επιλέγουμε ένα τυχαίο action αντί ενός action βασισμένο στον πίνακα Q. Αυτό συμβαίνει έτσι ώστε να μπορέσουμε να ανακαλύψουμε κάποια νέα πολιτική που ίσως να μην δοκιμάζαμε ποτέ σε περίπτωση που ο πίνακας Q αποκτούσε τιμές τέτοιες έτσι ώστε να κλείδωνε σε μια συγκεκριμένη πολιτική.

Παράμετροι α και γ:

Μεγάλο ρόλο έπαιξαν επίσης και οι δύο παράμετροι, α και γ, της συνάρτησης που ενημερώνει τον πίνακα Q.

- Η παράμερος α είναι ο ρυθμός μάθησης (learning rate). Όταν το α είναι 1, τότε ο πράκτοράς μας προτιμά την εξερεύνηση από το να επαναπαυτεί σε αυτά που έχει ήδη μάθει και οι τιμές του πίνακα Q ενημερώνονται σύμφωνα με τα νέα δεδομένα. Αντιθέτως, όταν το α είναι κοντά στο 0, τότε ο πράκτοράς μας προτιμά τις τιμές του πίνακα Q που έχει ήδη μάθει και αυτές δεν επηρεάζονται πολύ από τα νέα observations και rewards. Επομένως, τόσο οι δοκιμές όσο και η λογική υπαγορεύει να αρχίσουμε από μια μεγάλη τιμή του α, εφόσον ο πράκτοράς μας δεν έχει καμία προηγούμενη γνώση, και σιγά σιγά καθώς μαθαίνουμε και καταλήγουμε σε κάποια βέλτιστη πολιτική να μειώνεται σταδιακά το α μέχρι να τίνει ασυμπτωτικά στο μηδέν. Αυτό ακριβώς υλοποιήθηκε τελικά. Η παράμετρος α ξεκινάει από 1 και παραμένει 1 για τα πρώτα 100 επεισόδια, ύστερα γίνεται 0.9 για τα επόμενα 100 επεισόδια, έπειτα 0.81 για τα επόμενα 100 επεισόδια κ.ο.κ. Δηλαδή η συνάρτηση είναι a=0.9^k, με το k να ξεκινάει από 0 και να αυξάνεται κατά 1 κάθε 100 επεισόδια. Το 0.02 έχει οριστεί ως η ελάχιστη τιμή που μπορεί να πάρει η παράμετρος α.

- Η παράμετρος γ καθορίζει την αξία των μελλοντικών rewards. Όταν το γ είναι 1, τότε ο πράκτοράς μας αξιολογεί τα μελλοντικά rewards ακριβώς το ίδιο με το άμεσο reward. Αυτό σημαίνει ότι ένα καλό action του πράκτορά μας μετά από 10 steps αξιολογείται το ίδιο με το να γινόταν στο επόμενο step. Επομένως, είναι λογικό να θεωρήσουμε ότι μεγάλη τιμή του γ δεν βοηθάει στο Q-Learning. Όταν το γ είναι κοντά στο 0, τότε ο πράκτοράς μας θα αξιολογεί μόνο τα άμεσα rewards. Επομένως, χρειαζόμαστε ένα πολύ καλό reward system για να λειτουργήσει καλά το Q-Learning στην περίπτωση αυτή. Αυτό ακριβώς υλοποιήθηκε τελικά. Επιλέχτηκε σταθερό γ=0.1 και αλλάχτηκε το reward system με τρόπο που θα περιγραφεί παρακάτω. Σημειώνεται ότι το reward system αλλάζει μόνο το reward που υπολογίζεται από την συνάρτηση ενημέρωσης του πίνακα Q. Το reward που καθορίζει την επιλυσιμότητα του προβλήματος MountainCar-v0 παραμένει το ίδιο, αυτό που δίνεται από το environment.

<u>Reward System:</u>

Το default reward system δίνει -1 για κάθε κίνηση εκτός και αν αυτή έχει ως αποτέλεσμα position<-0.5 και velocity<0, δηλαδή εάν ανεβαίνουμε τον αριστερό λόφο.

Έπειτα από πολλές δοκιμές και πειραματισμούς με διαφορετικά μοντέλα και συντελεστές, το τελικό reward system ήταν συνδυασμός δύο θεωρήσεων. Η πρώτη θεώρηση που έγινε ήταν απλή. Στόχος μας είναι να φτάσουμε όσο το δυνατόν υψηλότερα στον δεξιό λόφο, δηλαδή να μεγιστοποιήσουμε το position του αμαξιού (απώτερος στόχος το position=0.5). Ορίστηκαν, λοιπόν, δύο μεταβλητές οι οποίες αρχικοποιούνται στο -0.4 στην αρχή κάθε επεισοδίου. Η πρώτη λέγεται max_height και είναι το μέγιστο ύψος δεξιού λόφου (max position) στο οποίο έχει φτάσει το αμάξι στο επεισόδιο αυτό. Η δεύτερη λέγεται prev_height και αποθηκεύει το δεύτερο μεγαλύτερο max_height του επεισοδίου. Επομένως, κάθε φορά που το position του αμαξιού είναι μεγαλύτερο του max_height, το reward αυξάνεται κατά 0.4 αφού πρακτικά είναι η καλύτερη επίδοσή μας. Επίσης, κάθε φορά που το position του αμαξιού είναι μεταξύ prev_height και max_height, τότε το reward αυξάνεται κατά 0.2 αφού είναι η δεύτερη καλύτερη επίδοσή μας.

Η δεύτερη θεώρηση που έγινε είχε να κάνει με τον περιορισμό ότι ένα επεισόδιο μπορεί να έχει μόνο 200 steps. Δοκιμάζοντας με παραπάνω steps, το παραπάνω reward system μπορούσε να επιλύσει το MountainCar-v0 σε ακριβώς 100 επεισόδια. Αλλά με τον περιορισμό ότι ένα επεισόδιο μπορεί να έχει μόνο 200 steps, το environment γίνεται reset πριν το αμάξι καταφέρει να αναπτύξει ικανή ταχύτητα για να ανέβει τον λόφο. Επομένως, το σκεπτικό ήταν ότι θέλουμε να αναπτύξουμε μεγάλη ταχύτητα όσο το δυνατόν γρηγορότερα. Για τον λόγο αυτό το αποτέλεσμα του παραπάνω reward πολλαπλασιάστηκε με την απόλυτη τιμή του velocity πολλαπλασιασμλένη επί 5. Δηλαδή για maximum velocity 0.07, το reward πολλαπλασιάζεται επί 0.35. Ενώ για μικρό velocity π.χ. 0.005, το reward πολλαπλασιάζεται επί 0.0025. Παρατηρούμε λοιπόν ότι με τον τρόπο αυτόν τιμωρούμε αποφασιστικά την έλλειψη ταχύτητας και δίνουμε κίνητρο στο αμάξι να αναπτύξει υψηλό velocity νωρίς και να προλάβει να ανέβει τον λόφο πριν την ολοκλήρωση των 200 steps του επεισοδίου.

Τέλος, το reward αυξάνεται κατά 1 σε περίπτωση που καταφέρουμε να ανεβούμε επιτυχώς τον λόφο. Επίσης, αυξάνεται ξανά κατά 1 εάν το καταφέρουμε αυτό σε λιγότερο από 110 steps.

Επιλυσιμότητα:

Τέλος, για να ελέγχουμε την επιτυχή επίλυση του προβλήματος MountainCar-v0, κρατάμε σε κάθε επεισόδιο το συνολικό reward και το εκτυπώνουμε. Το reward αυτό αποθηκεύεται σε μια λίστα συνολικού μεγέθους 100 θέσεων. Εάν το συνολικό άθροισμα της λίστας αυτής είναι μεγαλύτερο του -11000 σημαίνει ότι έχω average reward μικρότερο του -110 σε 100 συνεχόμενα επεισόδια. Άρα έχω λύσει επιτυχώς το πρόβλημα. Αντίστοιχη λίστα έχει φτιαχτεί για να αποθηκεύει τον αντίστοιχο αριθμό ενεργειών κάθε επεισοδίου. Και οι δύο λίστες είναι μεγέθους 100 θέσεων και τύπου FIFO.

**Αποτελέσματα και παρατηρήσεις:**

Η αρχική ενισχυτική μάθηση (Q-Learning) που εφαρμόστηκε στο MountainCar-v0 είχε νωθρά αποτέλεσμα. Αρχικά, το αμάξι δεν ανέβαινε τον λόφο και τις περισσότερες φορές κολλούσε στο κέντρο μην μπορώντας να αναπτύξει ικανή ταχύτητα. Με την σωστή διακριτικοποίηση καταστάσεων, αυτό το πρόβλημα ξεπεράστηκε. Εντοπίστηκε ότι χρειαζόμαστε πολλά διαφορετικά velocity states για να γνωρίζει ο πράκτοράς μας ακριβώς που βρισκόμαστε. Τα position state είναι επίσης σημαντικά αλλά όχι σε τέτοιο βαθμό που να αξίζει να έχουμε πολλά, μιας και επηρεάζουν τον χρόνο αρνητικά.

Αφού έγινε η σωστή διακριτικοποίηση, το αμάξι ανέβαινε τον λόφο αλλά χρειαζόταν πολλά steps εντός του επεισοδίου για να το πετύχει αυτό, περίπου **1000+ steps** και **1000+ επεισόδια**. Επομένως, αλλάχτηκαν οι παράμετροι του Q-Learning, α και γ, με σκοπό την βελτίωση του αλγορίθμου. Όσον αφορά το α, παρατηρήθηκε ότι είναι αναγκαίο να είναι κοντά στο 1 έτσι ώστε το αμάξι να μαθαίνει γρήγορα. Δοκιμάζοντας, λοιπόν, με α=0.9 και γ=1, τα steps εντός του επεισοδίου που χρειαζόμασταν για να ανέβουμε τον λόφο μειώθηκαν σε περίπου **350 steps** και **700 επεισόδια**.

Αλλάζοντας το α απο σταθερό αριθμό σε φθίνουσα συνάρτηση και βρίσκοντας τους κατάλληλους συντελεστές, υπήρξε βελτίωση και το πρόβλημα λυνόταν έπειτα από **200 steps** και **400 επεισόδια**. Δοκιμάζοντας να μειώσουμε το γ, παρατηρούσαμε ότι η βελτίωση ήταν από ανύπαρκτη ως ραγδαία ανάλογα του reward system που χρησιμοποιούσαμε (είχαν φτιαχτεί 3 με 4 reward systems τότε).

Επομένως, θέτοντας γ=0.1, αρχίσαμε να κάνουμε δοκιμές με σκοπό την εύρεση του βέλτιστου reward system. Το πρώτο reward system που δοκιμάστηκε ήταν το reward να είναι ανάλογο του position observation. Το δεύτερο ήταν να παίρνουμε θετικό reward όταν το position και το velocity observation επιστρέφουν ομόσημες τιμές, αφού έτσι μεγιστοποιούμε το ύψος. Το τρίτο ήταν το reward να είναι ανάλογο της απόλυτης τιμής του velocity observation. Το τέταρτο ήταν να παίρνουμε θετικό reward κάθε φορά που ξεπερνάμε την μεταβλητή max_height ή prev_height που αναλύθηκαν παραπάνω. Το τελικό reward system, όπως είπαμε και παραπάνω, ήταν ο πολλαπλασιασμός του τρίτου με κατάλληλο συντελεστή και έπειτα συνδυασμός αυτού με το τέταρτο.

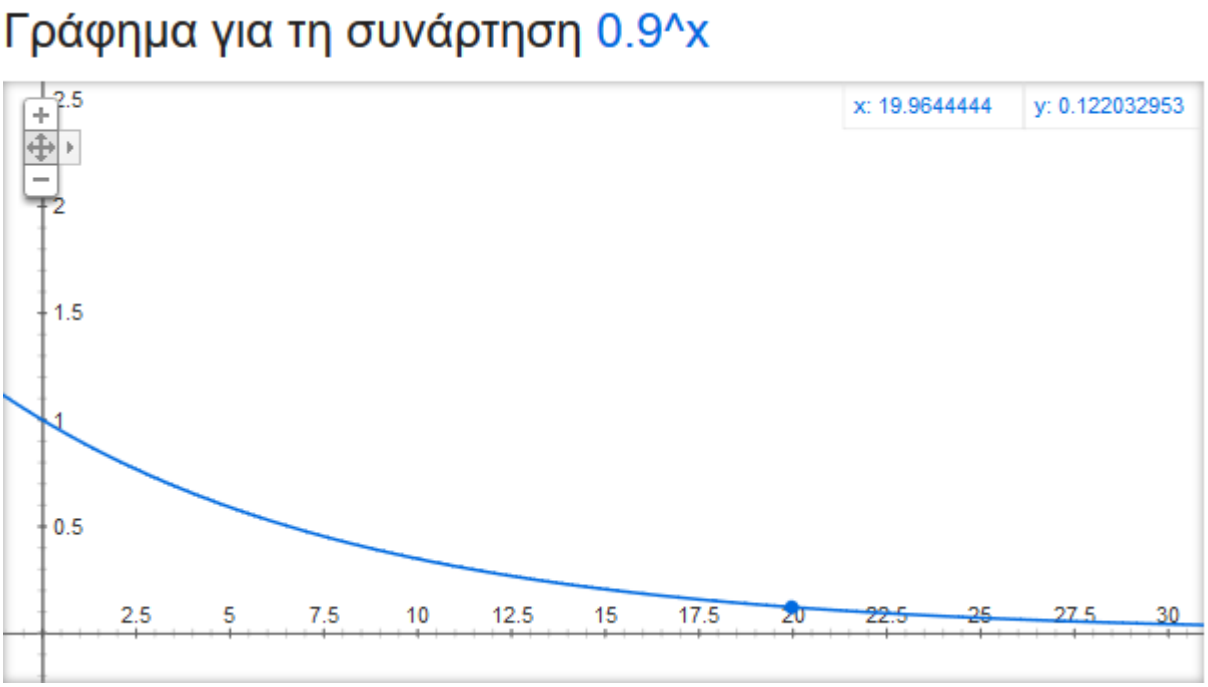
Με αυτό το reward system και με συντελεστές γ=0.1 σταθερό και α να καθορίζεται από φθίνουσα συνάρτηση, καταφέραμε να λύσουμε το πρόβλημα όπως ορίζει η βιβλιογραφία σε **200 steps** και **100 - 150 επεισόδια**. Το αποτέλεσμα αυτό, πιστεύω ότι είναι πολύ καλό αφού πρώτον, τα αντίστοιχα επεισόδια στο leaderboard ξεπερνούν τα 1000 και δεύτερον, έχει προστεθεί από την πλευρά μας η επιπλέον δυσκολία που

περιγράφηκε παραπάνω, δηλαδή ότι τα μη επιτυχή επεισόδια δίνουν σταθερά -200 reward (αντί για περίπου -160 που ίσως να έδιναν σύμφωνα με την βιβλιογραφία).

Αλλαγές:

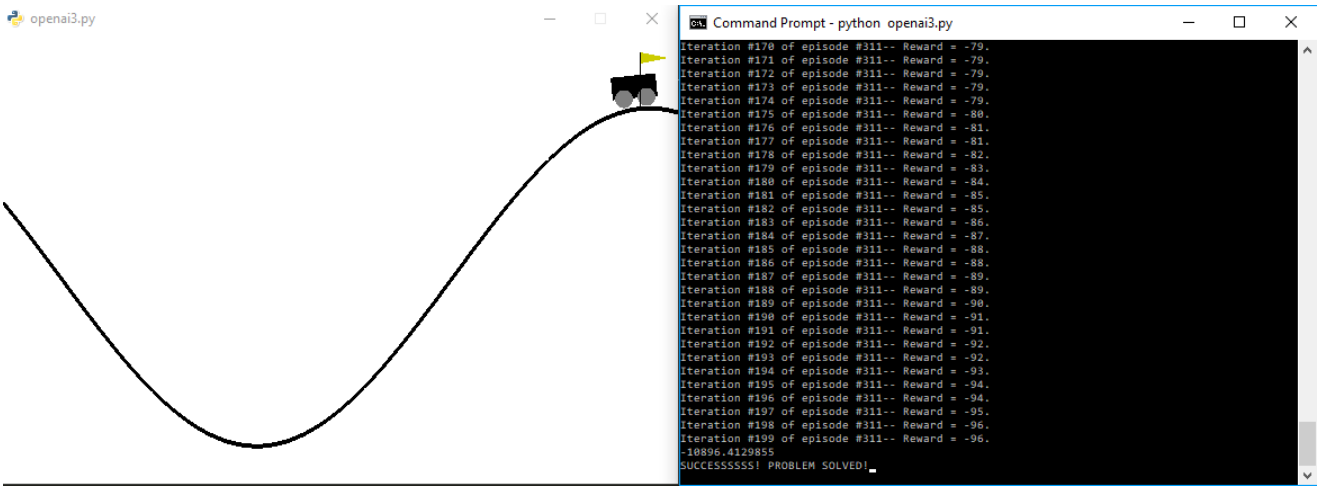
Με αφορμή την παράταση που δόθηκε, αλλάχτηκε ο κώδικας έτσι ώστε τα max_height και prev_height να μην είναι μεταβλητές που αρχικοποιούνται σε κάθε επεισόδιο αλλά να διατηρούν την τιμή τους για πάντα. Αυτό είχε ως αποτέλεσμα να καθυστερήσει η επίλυση του προβλήματος η οποία τώρα είναι περίπου στα **150 με 250 επεισόδια**. Αλλά ο παραγόμενος πίνακας Q είναι πολύ πιο αξιόπιστος και σε μερικά επεισόδια πετυχαίνει λύσεις με reward της τάξης του -60. Για να φανεί αυτό καλύτερα, στο τέλος του κώδικα προστέθηκε μια for loop μέσα στην οποία τρέχουμε το MountainCar-v0 για ακριβώς 100 επεισόδια σύμφωνα με τον πίνακα Q που έχουμε μάθει. Ως αποτέλεσμα, έχουμε επιτυχή επίλυση του προβλήματος με **average reward μεταξύ -80 και -110**.

Παρακάτω παραθέτω μια φωτογραφία που απεικονίζει την γραφική παράσταση της φθίνουσας συνάρτησης της παραμέτρου α. Υπενθυμίζεται ότι κάθε μονάδα του x ισοδυναμεί με 100 επεισόδια.



Επιπλέον παρατίθονται δύο φωτογραφίες που απεικονίζουν το αποτέλεσμα αρχικά (πρώτη φωτογραφία) με γ=1 και μη τελειοποιημένο reward system, και έπειτα (δεύτερη φωτογραφία) με γ=0.1 και το τελικό μας reward system. Επισημαίνεται ότι παρατηρήθηκε ότι η ενισχυτική μάθηση Q-Learning είναι ιδιαίτερα ευαίσθητη σε οποιαδήποτε αλλαγή, όσο μικρή κι αν είναι. Τα αποτελέσματα μπορούν να αλλάξουν δραματικά ακόμα. Εφόσον, λοιπόν, όσες παραπάνω αλλαγές και προσαρμογές έγιναν δεν οδήγησαν σε καλύτερα αποτελέσματα, καταλήγουμε στην υπάρχουσα λύση ως βέλτιστη. Μας εξασφαλίζει ανάβαση του λόφου εντός 40 επεισοδίων και γενική επίλυση του προβλήματος σε 100 με 200 επεισόδια.

Φωτογραφία 1 – Επίλυση μετά από 311 επεισόδια. Με μεγάλο γ.




Φωτογραφία 2 – Επίλυση μετά από 133 επεισόδια. Επίδραση του reward system.