ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

ΕΙΣΗΓΗΤΗΣ: ΠΑΝΑΓΙΩΤΗΣ ΚΙΚΑΣ

ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ: ΕΚΠΑΙΔΕΥΣΗ ΠΡΑΚΤΟΡΑ ΓΙΑ ΤΗΝ ΕΚΜΑΘΗΣΗ ΠΑΙΧΝΙΔΙΩΝ ΜΕΣΩ ΤΗΣ ΕΝΙΣΧΥΤΙΚΗΣ ΜΑΘΗΣΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων Καθηγητής: ΙΩΑΝΝΗΣ ΡΕΦΑΝΙΔΗΣ

30/06/2019

ΕΥΧΑΡΙΣΤΙΕΣ

Στο σημείο αυτό θα ήθελα να ευχαριστήσω όλους τους ανθρώπους που με βοήθησαν στην εκπόνηση αυτής της πτυχιακής εργασίας. Καταρχήν θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της εργασίας κ. Ιωάννη Ρεφανίδη για τη βοήθεια και την υποστήριξη που μου πρόσφορε όσες φορές χρειάστηκα να λύσω απορίες. Πιστεύω ότι οι γνώσεις και η εμπειρία που πήρα από τη συγκεκριμένη εργασία είναι ανεκτίμητης αξίας και τον ευχαριστώ που μου επέτρεψε να ασχοληθώ με ένα θέμα που με ενδιαφέρει τόσο πολύ.

Έπειτα θα ήθελα να ευχαριστήσω όλα τα κοντινά μου άτομα που με υποστήριξαν και πίστεψαν σε εμένα όσες φορές το χρειαζόμουν. Χωρίς αυτούς η συγκεκριμένη πτυχιακή δεν θα είχε πραγματοποιηθεί.

Τέλος, την Nvidia για τη παροχή της Titan Xp που χρησιμοποιήθηκε σε αυτή την έρευνα για την εκπαίδευση του πράκτορα. Την ευχαριστούμε για τη πολύτιμη χορηγία της στην εκπόνηση αυτής της έρευνας.

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία έχει ως στόχο την διερεύνηση του ενισχυτικού τρόπου μάθησης στο κλάδο της μηχανική μάθησης με στόχο την εκπαίδευση ενός πράκτορα, ώστε να μάθει να πραγματοποιεί μια σειρά από σύνθετους στόχους . Ο πράκτορας εκπαιδεύτηκε σε διαφορετικά περιβάλλοντα παιχνιδιών ‘Atari’ με σκοπό να εξασφαλίσει το μέγιστο δυνατό όφελος σε καθένα από αυτά. Το όφελος αυτό, εξαρτάται κάθε φορά από τη φύση του περιβάλλοντος και ο πράκτορας θα πρέπει να ανακαλύψει τις κινήσεις που πετυχαίνουν καλύτερα το στόχο του.

Παρουσιάζονται δύο διαφορετικά είδη ενισχυτικής μάθησης και τα αποτελέσματα που έδωσε το καθένα σε διαφορετικά περιβάλλοντα.

Η πρώτη μέθοδος ονομάζεται Deep Q Learning και προσπαθεί να προσεγγίσει τη πραγματική συνάρτηση χρησιμότητας και ύστερα από αυτή συμπεραίνει τη πολιτική που θα χρησιμοποιήσει.

Η δεύτερη μέθοδος ονομαζόμενη Actor Critic και πραγματοποιεί παραμετροποίηση στην ίδια τη πολιτική και προσπαθεί να τη βελτιστοποίηση. Αυτή η μέθοδος περιέχει και στοιχεία από τη προηγούμενη μέθοδο εκπαίδευσης και για αυτό θα παρουσιαστεί στη συνέχεια.

Για κάθε μια από αυτές τις μεθόδους παρουσιάζεται μια περιγραφή του τρόπου λειτουργίας, τα πλεονεκτήματα και τα μειονεκτήματα καθώς και η απόδοση τους σε διάφορα περιβάλλοντα εκπαίδευσης.

Τέλος, να σημειωθεί ότι και οι δυο αλγόριθμοι εκπαίδευσης παρέμειναν ίδιοι για όλα τα περιβάλλοντα εκπαίδευσης με τις μόνες αλλαγές στις παραμέτρους. Αυτό σημαίνει ότι η μάθηση δεν εξαρτάται από την φύση του περιβάλλοντος και η ίδια τεχνική μάθησης μπορεί να χρησιμοποιηθεί σε οποιοδήποτε περιβάλλον περιέχει σύνθετους στόχους και μια συνάρτηση αναμενόμενης ωφέλειας που ο πράκτορας πρέπει να μεγιστοποιήσει.

Λέξεις Κλειδιά : Ενισχυτική μάθηση, Συνάρτηση ωφέλειας, Markov Decision Processes (MDP’S), Μάθηση Atari, Προσέγγιση Συνάρτησης Χρησιμότητας, Asynchronous Advantage Actor-Critic(A3C)

ABSTRACT

With this thesis I try to get into depth to the research of Reinforcement Learning in the field of Machine Learning. The goal is to train an agent on a series of task with composite goals. The agent was trained on different Atari environments with the goal to achieve the optimal return. The return depends on the structure of each environment and the agent should discover the actions that achieve optimal behavior.

In this thesis I present two different methods of Reinforcement Learning and the results that they achieved on different environments.

The first method is called Deep Q Learning and It strives to approximate the optimal value function and then behave optimally according to it.

The second one is called Actor-Critic and it parametrizes the policy as a function and tries to optimize it. This method envelopes elements from the previous method and so It will be presented in the second part of this thesis.

For each method I describe how they operate from the inside, their advantages and disadvantages and their efficiency on the various tasks.

Last but not least, both algorithms remain unchanged for all the environments, meaning that they are not defendant on the structure of the environment and can be used to achieve any task.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

1 Εισαγωγή

Ο κλάδος της μηχανικής μάθησης είναι από τους πιο δημοφιλής κλάδους στις μέρες μας. Αμέτρητες επιχειρήσεις επενδύουν μεγάλα ποσά στο συγκεκριμένο κλάδο με σκοπό την έρευνα και ανάπτυξή του. Αυτό είχε ως συνέπεια την πληθώρα από εφαρμογές και τεχνολογίες που βασίζονται στη μηχανική μάθηση και τα νευρωνικά δίκτυα

Κάποια παραδείγματα είναι η εφαρμογή της σχεδόν σε όλα τα Κοινωνικά Δίκτυα με σκοπό την παρουσίαση εξατομικευμένου περιεχομένου (Youtube, Amazon, Netflix), στο κλάδο του Marketing για τη συσταδοποίηση πελατών με τα ίδια χαρακτηριστικά και τάσεις, στο κλάδο της επιστήμης για την εκλογή διαφόρων συμπερασμάτων (για παράδειγμα για την αναγνώριση μιας ασθένειας) και πολλά ακόμη.

Αυτή η συνεχόμενη τάση δείχνει ότι η μηχανική μάθηση γίνεται ολοένα και περισσότερο ένα αναπόσπαστο κομμάτι της καθημερινότητας μας και αυξάνει την ανάγκη για την παραπέρα έρευνά της. Οι δυνατότητες που μπορεί να μας προσφέρει είναι απαράμιλλής σημασίας δίνοντας στα προγράμματα που αναπτύσσονται μια νέα είδους ‘νοημοσύνη΄: τη δυνατότητα να μαθαίνουν και να γίνονται συνεχώς καλύτερα στο σκοπό τους.

Ο κλάδος της μηχανικής μάθησης χωρίζεται σε τρείς υποκατηγορίες με βάση το τρόπο με τον οποίο γίνεται η εκπαίδευση. Αυτές είναι:

- Επιβλεπόμενη μάθηση (Supervised Learning)
- Μη-Επιβλεπόμενη μάθηση (Unsupervised Learning)
- Ενισχυτική μάθηση (Reinforcement Learning)

Η παρούσα εργασία ασχολείται με το τρίτο είδος μάθησης, την Ενισχυτική. Σε αντίθεση με τις πρώτες δύο μεθόδους όπου χρειάζονται ένα μεγάλο όγκο από έτοιμα παραδείγματα εκπαίδευσης, η ενισχυτική μάθηση συλλέγει τα αντίστοιχα παραδείγματα εκπαίδευσης την ώρα που πραγματοποιεί την εργασία που του έχει δοθεί. Ταυτόχρονα λαμβάνει σε τακτά χρονικά διαστήματα ένα βραβείο ή

μια τιμωρία ανάλογα με τη πρόοδό του και βελτιώνει συνεχώς τη συμπεριφορά του ώστε να πάρει τα περισσότερα ‘βραβεία’.

Η ενισχυτική μάθηση αναπτύχτηκε αρχικά με βάση το πώς μαθαίνει ο ανθρώπινος εγκέφαλος μέσω δηλαδή διαφόρων σημάτων από το περιβάλλον σε μορφή ‘ανταμοιβών ή ‘τιμωρίας’. Μετά από πολλές επαναλήψεις οι άνθρωποι μαθαίνουν ποιες ενέργειες ή σειρά ενεργειών τους οδηγεί να λαμβάνουν τις περισσότερες ΄ανταμοιβές και προσαρμόζουν τη συμπεριφορά τους κατάλληλα.

Εκτός από τη δυνατότητα μιας βαθύτερης αντίληψης για το πώς μαθαίνει ο ανθρώπινος εγκέφαλος η ενισχυτική μάθηση μπορεί να μας βοηθήσει να αναπτύξουμε εφαρμογές που μαθαίνουν από το περιβάλλον στις οποίες οι κλασσικές μέθοδοι επιβλεπόμενης μάθησης είναι πολύ δαπανηρές ή ως και αδύνατες.

Η ενισχυτική μάθηση αναπτύχθηκε ραγδαία όταν η DeepMind εξέδωσε ένα άρθρο με τίτλο ‘Παίζοντας παιχνίδια Atari με τη χρήση της ενισχυτική μάθησης’. Το συγκεκριμένο άρθρο εκτόξευσε την έρευνα της ενισχυτική μάθησης στα ύψη και παρουσίασε τις απίστευτες δυνατότητες που παρέχει. Εκείνο το καιρό η εκπαίδευση πρακτόρων για να παίζουν παιχνίδια Atari ήταν κάτι ακατανόητο και άνοιξε τις πόρτες σε ένα νέο κλάδο.

Η παρούσα πτυχιακή ακολουθεί τα βήματα της Deepmind με στόχο την ανάπτυξη και την εφαρμογή αλγορίθμων ενισχυτικής μάθησης σε διάφορα περιβάλλοντα Atari. Αρχικά όμως πρέπει να παρουσιαστούν κάποια θεμελιώδης στοιχεία σε σχέση με την ενισχυτική μάθηση.

2 Ενισχυτική Μάθηση

2.1 Γενικά

Όπως αναφέρθηκε η μεγαλύτερη διαφοροποίηση της ενισχυτικής μάθησης από τις υπόλοιπες μεθόδους της μηχανικής μάθησης είναι η έλλειψη ενός επόπτη από τη διαδικασία της εκπαίδευσης. Το μόνο που χρειάζεται για την εκπαίδευση του πράκτορα είναι ένα σήμα που μεταφέρει στο πρόγραμμα το πόσο καλά τα πάει σε μορφή ανταμοιβής / τιμωρίας.

Το βασικό μοντέλο στην ενισχυτική μάθηση περιλαμβάνεται από δυο βασικά στοιχεία: Το περιβάλλον και τον πράκτορα. Το περιβάλλον τροφοδοτεί αρχικά τον πράκτορα με μια αρχική κατάσταση/παρατήρηση. Ο πράκτορας αφού επεξεργαστεί την παρατήρηση αυτή πραγματοποιεί μια ενέργεια A. Το περιβάλλον στη συνέχεια ανάλογα με το πόσο καλή ήταν η ενέργεια που έκανε του στέλνει την ανταμοιβή του R και τη νέα κατάσταση του περιβάλλοντος. Αυτός ο κύκλος επαναλαμβάνεται μέχρι ο πράκτορας να λύσει το πρόβλημα που του ζητείτε ή να ενεργοποιηθεί κάποια συγκεκριμένη συνθήκη στο περιβάλλον.

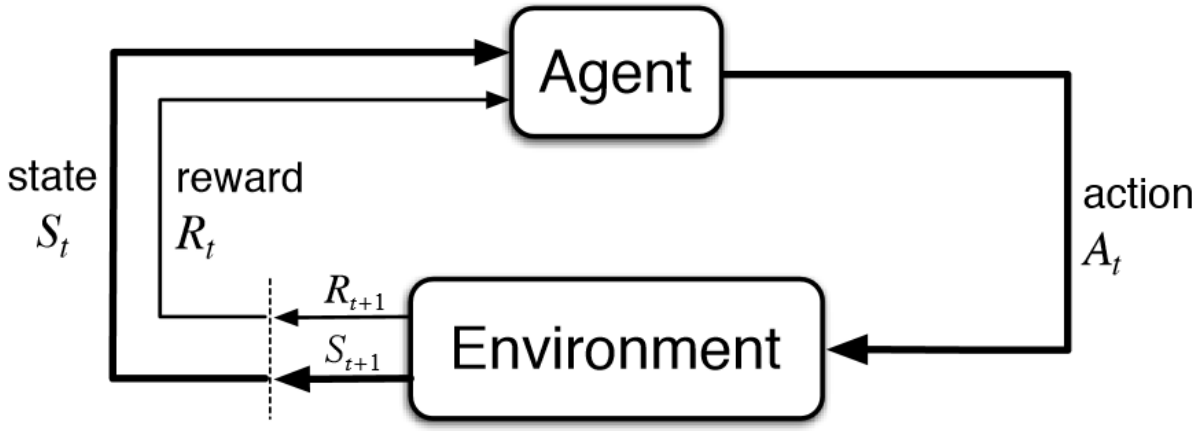


FIGURE 1 ΒΑΣΙΚΟ ΜΟΝΤΕΛΟ ΕΝΙΣΧΥΤΙΚΗΣ ΜΑΘΗΣΗΣ

Αρκετές φορές η ανταμοιβή μπορεί να έρχεται με καθυστερήσεις. Αυτό δυσκολεύει το πράκτορα και δημιουργεί το λεγόμενο 'Πρόβλημα απόδοσης Ευθηνών'. Είναι δύσκολο για το πρόγραμμα να καταλάβει από ένα μεγάλο αριθμό κινήσεων ποιες ήταν αυτές που των οδήγησαν σε ένα καλό ή κακό

αποτέλεσμα ώστε να τις επαναλάβει περισσότερο ή να τις ελαττώσει. Η λύση σε αυτό το πρόβλημα είναι ο μεγάλος αριθμός επαναλήψεων μέχρι να γίνει αυτός ο διαχωρισμός των ‘καλών’ από των ‘κακών’ κινήσεων.

Επίσης, ο πράκτορας θα πρέπει να μεγιστοποιήσει το μέγιστο συνολικό αναμενόμενο όφελος του. Αυτό συνεπάγεται πολλές φορές τη θυσία μιας άμεσης ανταμοιβής με σκοπό την απόκτηση μεγαλήτερης στο μέλλον. Επομένως, καταλαβαίνουμε ότι ο χρόνος έχει σημασία σε αντίθεση με άλλες μεθόδους μάθησης και η μάθηση γίνεται με ακολουθιακό τρόπο.

Η εκπαίδευση γίνεται σε επεισόδιά τα οποία είναι μια ακολουθία κινήσεων και τερματίζονται με την πραγματοποίηση του στόχου ή με την ικανοποίηση μιας συνθήκης του περιβάλλοντος. Έτσι μπορούμε να πούμε ότι όλοι οι στόχοι ενός πράκτορα μπορούνε να περιγράφουνε από την μεγιστοποίηση του αναμενόμενου οφέλους στη διάρκεια ενός επεισοδίου. Τελος, θα αναφερομε στην πραγματοποίηση μιας ενέργειας με τον όρο ‘βημα’.

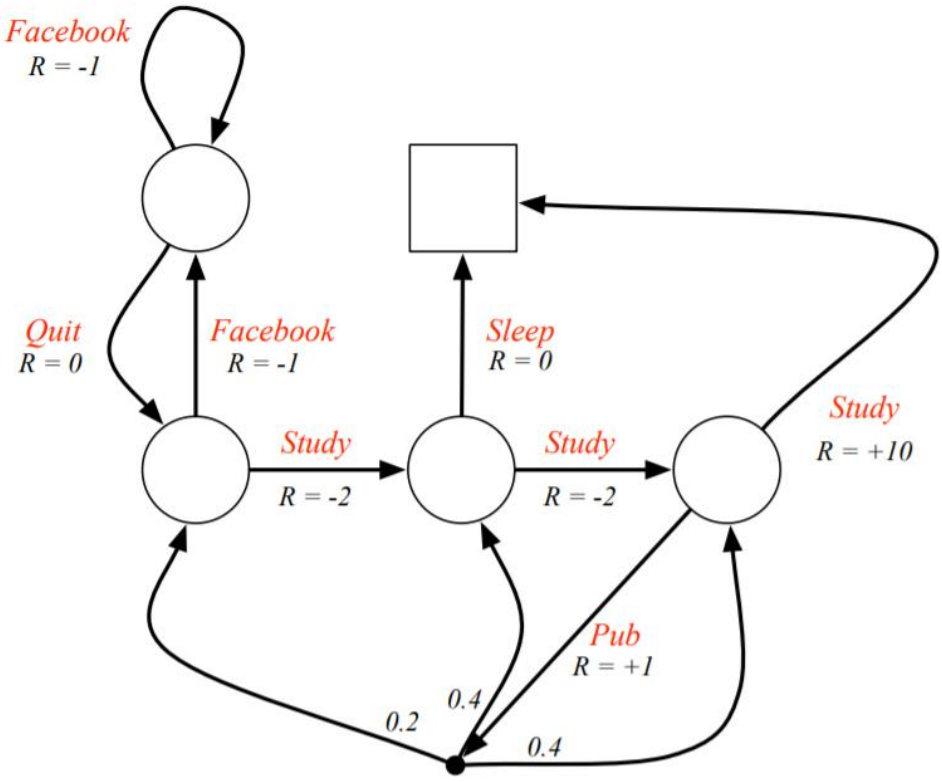
2.2 Διαδικασίες Απόφασης Markov (MDP's)

Μια κατάσταση ονομάζεται ως κατάσταση 'Markov' αν περιέχει όλες τις σημαντικές πληροφορίες από τις παρελθοντικές καταστάσεις δηλαδή είναι ένα επαρκές στατιστικό του παρελθόντος. Αυτό σημαίνει ότι αν ένας πράκτορας βρίσκεται σε μια κατάσταση Markov μπορεί να λάβει αποφάσεις για το μέλλον μονό με βάση αυτή και δεν χρειάζεται να κρατάει ιστορικό από όλες τις καταστάσεις που έχουν προηγηθεί. Δηλαδή αν με S_t συμβολίσουμε τη κατάσταση του πράκτορα στη χρονική στιγμή t τότε μια κατάσταση λέγεται Markov αν και μόνο αν:

$$P[S(t+1)|S(t)] = P[S(t+1)|S(1), S(2), \dots, S(t)]$$

Όλα τα περιβάλλοντα με τα οποία θα ασχοληθούμε μπορούνε να περιγραφούνε ως διαδικασίες απόφασης Markov. Θα συμβολίσουμε με O_t την παρατήρηση του περιβάλλοντος στη χρονική στιγμή t, A_t την ενέργεια που πραγματοποιεί και R_t την άμεση ανταμοιβή που λαμβάνει από αυτή την ενέργεια. Σε όλα τα περιβάλλοντα ως κατάσταση θα θεωρήσουμε την παρατήρηση του περιβάλλοντός του O_t και με βάση αυτή μόνο ο πράκτορας βγάζει συμπεράσματα για το ποια κίνηση να ακολουθήσει.

Σε μια κατάσταση Markov θα συμβολίσουμε με $P_{ss'}$ τη πιθανότητα μετάβασης από τη κατάσταση s στην s'. Αυτή η πιθανότητα εξαρτάται από τη δυναμική και τη φύση του κάθε περιβάλλοντος. Κάθε διαδικασία ή αλυσίδα απόφασης αποτελείται από ένα σύνολο καταστάσεων S και μια πιθανότητα μετάβασης $P_{ss'}$ για κάθε κατάσταση. Αν επιπλέον προσθέσουμε και ένα σύνολο από ενέργειες που μπορεί να πραγματοποιήσει ο πράκτορας και μια συνάρτηση ανταμοιβής Rs^a με βάση τη κατάσταση στην οποία βρίσκεται και την ενέργεια που πραγματοποιεί παίρνουμε το πλήρη ορισμού των διαδικασιών απόφασης Markov. Ένα παράδειγμα είναι το παρακάτω όπου δείχνει μια διαδικασία απόφασης Markov για έναν μαθητή:



Στο παραπάνω σχήμα οι καταστάσεις αναπαρίστανται με κύκλο. Τα τετράγωνά αναπαριστούν ότι μια κατάσταση είναι τελική, δηλαδή αν φθάσει εκεί τελειώνει το επεισόδιο. Επίσης, σε κάθε βέλος με το σύμβολο R συμβολίζεται η άμεση ανταμοιβή από την ολοκλήρωση της ενέργειας. Τέλος, ο μαύρος κύκλος στο κάτω μέρος συμβολίζει ότι δεν υπάρχει κάποια σίγουρη κατάσταση στην οποία θα μεταβεί ο πράκτορας με την εκτέλεση της ενέργειας αλλά εξαρτάται από τις αντίστοιχές πιθανότητες μετάβασης που υπάρχουν στο κάθε βέλος. Στις υπόλοιπες ενέργειες ο πράκτορας έχει πιθανότητα μετάβασης ίση με 1 και για αυτό δεν αναγράφονται.

2.3 Συνάρτηση ωφέλειας και Εξισώσεις Bellman

Ως G_t θα συμβολίσουμε το συνολική επιστροφή από τη χρονική στιγμή t ως το τέλος του επεισοδίου. Δηλαδή ισχύει ότι:

$$G_t = R_t + \gamma * R_{t+1} + \ldots = \sum \gamma^k R_{t+k}$$

Οι μελλοντικές ανταμοιβές προ εξοφλούνται με έναν ρυθμό γ ο οποίος ονομάζεται παράγοντας προεξόφλησης. Ο λόγος που γίνεται αυτό είναι γιατί έτσι αναπαριστάμε καλύτερα την αβεβαιότητα που υπάρχει για το μέλλον, γίνεται αποφυγή ατέρμονων βρόχων, καθώς και γιατί σε πολλές περιπτώσεις η άμεση ανταμοιβή έχει μεγαλύτερη βαρύτητα από τις μελλοντικές. Ο παράγοντας προεξόφλησης γ παίρνει τιμές <= 1 και καθορίζει το πόσο σημαντικές είναι οι μελλοντικές ανταμοιβές. Αν γ = 0, ο πράκτορας θα επικεντρώνεται μόνο στις άμεσες ανταμοιβές ενώ αν γ = 1 ο πράκτορας έχει όλες τις μελλοντικές ανταμοιβές στην ίδια αξία όσο της άμεσης. Συνήθως επιλέγουμε μια τιμή μεταξύ 0.90 και 0.99.

Μια πολιτική 'π' είναι μια κατανομή πιθανοτήτων των ενεργειών που μπορεί να πραγματοποιήσει ο πράκτορας στη συγκεκριμένη κατάσταση:

$$\pi(a|s) = P[A_t=a \mid S_t=s]$$

Θα συμβολίσουμε με $v_\pi(s)$ ως συνάρτηση ωφέλειας μιας κατάστασης S δηλαδή την αναμενόμενη αθροιστική συνολική μελλοντική ανταμοιβή της κατάστασης S αν ακολουθεί πολιτική π:

$$v_\pi(s) = E[G_t \mid S_t = s]$$

Ως $q_\pi(s,a)$ θα συμβολίσουμε τη συνάρτηση ωφέλειας της κατάστασης αν πάρει την ενέργεια a ενώ ακολουθει πολιτική π, δηλαδή:

$$q_\pi(s,a) = E_\pi[G_t \mid S_t=s,\ A_t=a]$$

Αν αναλύσουμε τις παραπάνω δυο ποσότητες τότε βλέπουμε ότι μπορούνε να γραφούν ως:

$$v_\pi(s) = E_\pi[R_t + \gamma * v(s_{t+1})|S_t = s]$$
$$= \sum_{a \in A} \pi(s|a) * (R_s^a + \gamma * \sum_{s' \in S} P_{ss'}^a v_\pi(s'))$$

$$q_\pi(s,a) = E_\pi[R_{t+1} + \gamma * q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a]$$
$$= R_s^a + \gamma * \sum_{s' \in S} P_{ss'} \sum_{\alpha' \in A} \pi(\alpha'|s') q_\pi(s', a')$$

Αυτές οι δυο εξισώσεις λένε ουσιαστικά ότι η συνολική χρησιμότητα από τη κατάσταση ‘s’ ακολουθώντας πολιτική π ισούται με την άμεση ανταμοιβή και την προεξόφληση της συνάρτησης χρησιμότητας στην επόμενη κατάσταση. Αυτές οι εξισώσεις αναφέρονται και ως ‘Αναμενόμενες Εξισώσεις Bellman’.

Τις περισσότερες φορές θέλουμε να υπολογίσουμε ποια είναι η μέγιστη συνάρτηση χρησιμότητας για όλες τις πολιτικές που μπορεί να ακολουθήσει ο πράκτοράς μας από μια συγκεκριμένη κατάσταση. Οι επόμενες δυο ποσότητες εκφράζουν ακριβώς αυτό:

$$v_*(s) = \max_\pi v_\pi(s)$$

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

Φυσικά, αν ο πράκτορας έχει τη μέγιστη χρησιμότητα ενέργειας q_* για οποιαδήποτε κατάσταση s, τότε αρκεί να ακολουθεί κάθε φορά την ενέργεια που του δίνει το μεγαλύτερο q(s,a) ώστε να συμπεριφέρεται με το καλύτερο δυνατό τρόπο. Το πρόβλημα λοιπόν έγκειται στην εύρεση των παραπάνω δύο ποσοτήτων ώστε να βρεθεί η μέγιστη δυνατή πολιτική.

Οι παραπάνω δυο ποσότητες μπορούνε να αναπτυχθούν περισσότερο και να μας δώσουν:

$$v_*(s) = \max_a R_s^a + \gamma * \sum_{s' \in S} P_{ss'}^a v_*(s')$$

Αυτό μας λέει ότι η μέγιστη δυνατή χρησιμότητα αυτής της κατάστασης s ισούται με την μέγιστη άμεση ανταμοιβή και την προεξόφληση του αθροίσματος της μέγιστης χρησιμότητας όλων των πιθανών καταστάσεων s’ που μπορεί να βρεθεί επί τις αντίστοιχες πιθανότητες μετάβασης $P_{ss'}$.

Παρόμοια για τη μέγιστη δυνατή χρησιμότητα ενέργειας προκύπτει:

$$q_*(s,a) = R_s^a + \gamma * \sum_{s' \in S} P_{ss'}^a \max_{a'} q_*(s', a')$$

Ισούται δηλαδή με την ανταμοιβή από την ενέργεια a και το προ εξοφλούμενό άθροισμα των μέγιστων χρησιμοτήτων ενέργειας για όλες τις δυνατές καταστάσεις s’ επί τις αντίστοιχες πιθανότητές τους.

Οι παραπάνω δυο εξισώσεις λέγονται επίσης και ‘Εξισώσεις Bellman Μέγιστης Ωφέλειας’ (Bellman Optimality Equations) και πάνω σε αυτές βασίζονται οι περισσότερες σύγχρονοι μέθοδοι εκπαίδευσης ενισχυτικής μάθησης.

2.4 Μάθηση Χωρίς Μοντέλο

Σε περίπτωση που μας ήταν γνωστές οι δυναμικές του περιβάλλοντος δηλαδή όλες οι πιθανότητες μετάβασης $P_{ss'}$ για κάθε κατάσταση s τότε θα μπορούσαμε να επιλύσουμε το πρόβλημα της εύρεσης της μέγιστης δυνατής ωφέλειας με διάφορες μεθόδους του δυναμικού προγραμματισμού. Κάποιες από αυτές είναι:

- Επανάληψη πολιτικών και άπληστη βελτίωση τους
- Επανάληψη αξιών
- Αναζήτηση κατά βάθος

Γενικά όμως στα περισσότερα προβλήματα απόφασης Markov όπου θέλουμε να επιλύσουμε, το περιβάλλον που εκπαιδεύεται ο πράκτορας είναι άγνωστο ή πολύ περίπλοκο για να μοντελοποιηθεί. Θέλουμε λοιπόν ο πράκτορας μας να μάθει να συμπεριφέρεται με το καλύτερο δυνατό τρόπο μόνο με βάση την εμπειρία του από το περιβάλλον και χωρίς καμία γνώση για τις δυναμικές του.

Έχουν προταθεί διάφορες λύσεις στο πρόβλημα αυτό, δυο από τις κυριότερες οι οποίες είναι:

- Μέθοδος Προσέγγισης Συνάρτησης Αξιών
- Μέθοδος Κλίσης Πολιτικών

Η πρώτη προσπαθεί να προσεγγίσει τη μέγιστη συνάρτηση αξιών, και ύστερα προσαρμόζει τη πολιτική του λαμβάνοντας τις ενέργειες που δίνουν τη μεγαλύτερη συνολική χρησιμότητα. Η δεύτερη μέθοδος παραμετροποιεί την ίδια την πολιτική και προσπαθεί μέσω της κλασσικής μεθόδου της ανάβασης με βάση τη κλίση (Gradient Ascent) να βρει τις παραμέτρους της πολιτικής που δίνουν τη μεγαλύτερη χρησιμότητα.

Η πτυχιακή αυτή παρουσιάζει έναν ευρέως χρησιμοποιούμενο αλγόριθμο για κάθε κατηγορία καθώς και τα αποτελέσματα που έδωσαν. Οι αλγόριθμοι που θα παρουσιαστούν στα επόμενα κεφάλαια είναι:

- Deep Q learning (Off Policy, Προσέγγιση Συνάρτησης Αξιών)
- Ασύγχρονος Πλεονασματικός Actor Critic (Policy Gradient που συνδυάζει και κάποια χαρακτηριστικά της παραπάνω μεθόδου)

Τέλος μέσα στα χρόνια έχουν προταθεί και διάφορες μέθοδοι που χρησιμοποιούν γενετικούς αλγορίθμους και τεχνικές μετάλλαξης για να βρούνε τη μέγιστη συνάρτηση χρησιμότητας και διάφορες παραλλαγές αυτών. Η παρούσα πτυχιακή δεν τις καλύπτει και επικεντρώνεται στις παραπάνω μεθόδους.

2.5 Το πρόβλημα της αναζήτησης/ εκμετάλλευσης (Exploration / Exploitation)

Σε όλες τις μεθόδους της ενισχυτικής μάθησης υπάρχει το λεγόμενο δίλλημα της αναζήτησης ενάντια της εκμετάλλευσης και είναι το εξής:

Καθώς ο πράκτορας αναζητά να βρει τη βέλτιστη πολιτική στο συγκεκριμένο περιβάλλον έρχεται αντιμέτωπος με ολοένα και καλύτερες στρατηγικές. Το δίλημμα του έγκειται σε κάθε στιγμή στο αν θα συμπεριφερθεί με το βέλτιστο μέχρι τώρα τρόπο που έχει βρει (εκμετάλλευση) ή αν θα αναζητήσει το περιβάλλον για μια νέα πολιτική που πιθανώς θα αποδειχθεί καλύτερη από αυτή που ακολουθεί μέχρι τώρα. Η εκμετάλλευση και η αναζήτηση έχουν μια αντιστρόφως ανάλογη σχέση. Είναι σημαντικό να υπάρχουν και τα δυο σε επαρκή βαθμό.

- Αν εκμεταλλεύεται και βελτιώνει μόνο μια πολιτική που έχει βρει χωρίς να εξερευνάει για νέες τότε υπάρχει η πιθανότητα να καταλήξει σε ένα τοπικό μέγιστο και να μη βρει ποτέ τη βέλτιστη πολιτική
- Αν από την άλλη εξερευνάει συνεχώς καινούργιες πολιτικές και δεν εκμεταλλεύεται καμία τότε δεν θα βελτιώσει ποτέ επαρκώς κάποια και δεν θα φθάσει ποτέ στη βέλτιστη πολιτική

Καταλαβαίνουμε λοιπόν ότι θα πρέπει να βρούμε μια ισορροπία στην οποία ο πράκτορας βελτιώνει τις βέλτιστες πολιτικές που έχει βρει αλλά αφήνει χώρο για εξερεύνηση νέων.

Συνήθως η πολιτική που ακολουθούμε είναι να αρχίζουμε με ένα υψηλό ρυθμό εξερεύνησης (π.χ. 1 = τυχαίες κινήσεις) επειδή στην αρχή το περιβάλλον είναι άγνωστο και σταδιακή μείωση του ώστε ο πράκτορας να αρχίσει να βελτιώνει τις πολιτικές που έχει ανακαλύψει.

3 OpenAI Gym και άλλες Τεχνολογίες

3.1 Το OpenAi Gym

Τα περιβάλλοντα εκπαίδευσης προέρχονται από το OpenAI Gym. Το OpenAi Gym είναι ένα module που περιέχει διάφορα έτοιμα περιβάλλοντα εκπαίδευσης όπου μπορεί κάποιος να προπονήσει τους αλγόριθμούς ενισχυτικής μάθησης που έχει αναπτύξει. Το OpenAI Gym μας παρέχει έτοιμες τις παρατηρήσεις του πράκτορα για το περιβάλλον, μια συνάρτηση ανταμοιβής ανάλογα με το πόσο κοντά στους στόχους του περιβάλλοντος συμπεριφέρεται και διάφορες διαθέσιμες ενέργειες που μπορεί να ετκελέσει. Επίσης, ενημερώνει αυτόματα κάθε φορά που γίνεται μια κίνηση το περιβάλλον και στέλνει πίσω τη νέα παρατήρηση του στον πράκτορα. Έτσι μας βοηθάει να επικεντρωθούμε στη σχεδίαση των αλγορίθμων ενισχυτικής μάθησης χωρίς να ανησυχούμε για τους μηχανισμούς του περιβάλλοντος.

Κάθε περιβάλλον στο OpenAI Gym έχει διαφορετικές παρατηρήσεις και πλήθος από δυνατές ενέργειες. Οι παρατηρήσεις ή καταστάσεις του πράκτορα, είναι ουσιαστικά οι είσοδοι του νευρωνικού δικτύου βάση του οποίου ο πράκτορας πρέπει να αποφασίζει ποιες ενέργειες να πραγματοποιήσει. Η έτοιμη συνάρτηση env.observation_space μας επιστρέφει το μέγεθος της εισόδου / παρατηρήσεων και η συνάρτηση env.action_space.n το πλήθος των δυνατών ενεργειών. Οι ενέργειες συμβολίζονται ως αριθμοί ξεκινώντας από το μηδέν και αυξάνοντας.

Το OpenAI Gym μας παρέχει 4 βασικές συναρτήσεις για την εκπαίδευση:

- make(): Με παράμετρο το όνομα του περιβάλλοντος που θέλουμε να δημιουργήσουμε, φτιάχνει το περιβάλλον και μας επιστρέφει μια αναφορά σε αυτό. Μια λίστα όλων των διαθέσιμων περιβαλλόντων που μας παρέχει το OpenAI Gym μπορεί να βρεθεί εδώ: https://gym.openai.com/envs/#classic_control
- env.reset(): Αυτή η μέθοδος φέρνει το περιβάλλον στην αρχική του κατάσταση και μας επιστρέφει μια παρατήρηση για αυτό. Η συγκεκριμένη συνάρτηση εκτός από την αρχή της εκπαίδευσης θα πρέπει επίσης να καλείτε κάθε φορά που τελειώνει ένα επεισόδιο.

- env.step('action'): Αυτή η συνάρτηση παίρνει ως παράμετρο μια ενέργεια από τις διαθέσιμες ενέργειες, δηλαδή έναν αριθμό από το 1 έως το env.observation.n και τη πραγματοποιεί. Έπειτα αφού υπολογίσει τη νέα κατάσταση του περιβάλλοντος επιστρέφει τέσσερις παραμέτρους: next_state, reward, done, info. Η next_state είναι η νέα παρατήρηση ή κατάσταση δηλαδή του περιβάλλοντος μετά την πραγματοποίηση της ενέργειας. Η επιστρεφόμενη μεταβλητή reward είναι ένας αριθμός και συμβολίζει την ανταμοιβή που έλαβε ο πράκτορας πραγματοποιώντας αυτήν την ενέργεια. Η done είναι μια λογική μεταβλητή και δείχνει με τιμή true ή false αν έχει τελειώσει το επεισόδιο. Κάθε περιβάλλον έχει διαφορετικές συνθήκες για το πότε θεωρείτε ότι έχει φθάσει στο τέλος του το επεισόδιο. Τέλος, η μεταβλητή info περιέχει διάφορα διαγνωστικά στατιστικά, χρήσιμα για debugging. Δεν θα πρέπει να χρησιμοποιηθεί ως βοήθεια από τον πράκτορα στη διαδικασία μάθησης.
- env.render(): Εμφανίζει το περιβάλλον σε ένα παράθυρο. Αυτή η συνάρτηση δεν θα πρέπει να χρησιμοποιείτε στη φάση της εκπαίδευσης παρά μόνο στη φάση του ελέγχου γιατί καθυστερεί αρκετά το περιβάλλον.
- env.close(): Απελευθερώνει την αναφορά στο περιβάλλον που χρησιμοποιεί. Στο τέλος κάθε εκτέλεσης θα πρέπει πάντα να καλείτε.

Στη συνέχεια παρουσιάζονται τα περιβάλλοντα που χρησιμοποιήθηκαν για την παρούσα πτυχιακή εργασία.

3.2 Περιβάλλοντα Εκπαίδευσης

Οι πράκτορες εκπαιδεύτηκαν σε τέσσερα περιβάλλοντα εκπαίδευσης. Παρακάτω γίνεται μια περιγραφή του καθενός: Το μέγεθος της παρατήρησης, οι διαθέσιμες ενέργειες, οι στόχοι του πράκτορα στο καθένα και πότε θεωρείτε ότι έχει τερματίσει το επεισόδιο.

3.2.1 CartPole-v0

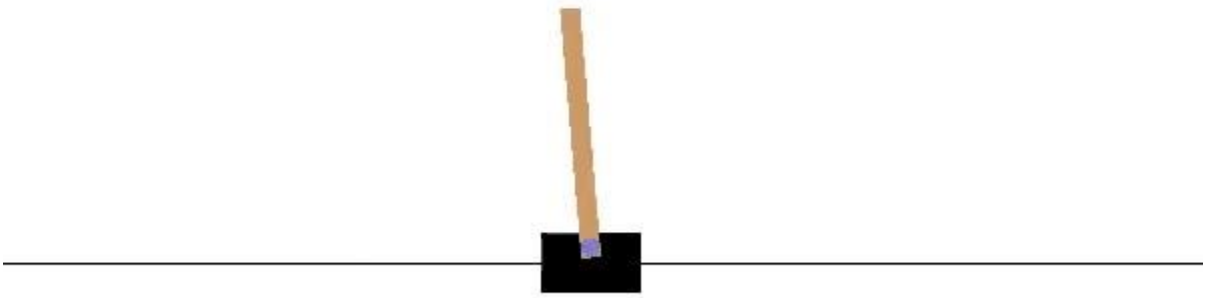


FIGURE 2 ΑΠΛΟ CARTPOLE ΠΕΡΙΒΑΛΛΟΝ

Σε αυτό το περιβάλλον ο πράκτορας ελέγχει το μαύρο βαγόνι με το στύλο από πάνω του και έχει ως στόχο να μάθει μια στρατηγική που θα του επιτρέπει να κρατάει το στύλο όρθιο για όσο περισσότερη ώρα γίνεται. Πιο συγκεκριμένα:

- Οι παρατηρήσεις ή κατάσταση που έχει ο πράκτορας ως είσοδο είναι μεγέθους τέσσερα και αντιπροσωπεύουν: τη θέση του βαγονιού σε σχέση με τη μέση, όπου η μέγιστη θέση που μπορεί να έχει είναι μεταξύ [-4.8, 4.8], τη ταχύτητα του, τη γωνία που σχηματίζει ο στύλος με το βαγόνι και τη ταχύτητα του στύλου.

- Οι δυνατές ενέργειες που μπορεί να πραγματοποιήσει ο πράκτορας είναι δύο και είναι: Να σπρώξει το βαγόνι με δύναμη +-1 προς τα δεξιά ή τα αριστερά αντίστοιχα.
- Η ανταμοιβή που παίρνει είναι +1 σε κάθε βήμα και για αυτό πρέπει να κρατήσει το στύλο όρθιο όσο περισσότερη ώρα γίνεται.
- Αρχική Κατάσταση/Παρατήρηση: Οι τέσσερις είσοδοι παίρνουνε μια τυχαία τιμή από το διάστημα [-0.05, 0.05]
- Το επεισόδιο θεωρείτε ότι έχει τερματίσει αν έστω μία από τις παρακάτω τρείς συνθήκες ικανοποιηθεί: Ο στύλος έχει σχηματίσει γωνία μεγαλύτερη των 12 μοιρών, η απόσταση του βαγονιού από το κέντρο είναι μεγαλύτερη από +- 2.4 που σημαίνει ότι βγήκε έξω από τα όρια της οθόνης και τέλος όταν το επεισόδιο ξεπεράσει τα 200 βήματα.
- Το περιβάλλον θεωρείτε λυμένο αν η μέση ανταμοιβή είναι μεγαλύτερη ή ίση του 195 για 100 συνεχόμενα επεισόδιά.

Το περιβάλλον αυτό είναι αρκετά απλοϊκό και χρησιμοποιήθηκε για να ελεγχθεί σε πρώτη φάση αν οι αλγόριθμοι που υλοποιήθηκαν εκπαιδεύονται και δεν περιέχουν κάποιο σφάλμα.

3.2.2 PongNoFrameskip-v4

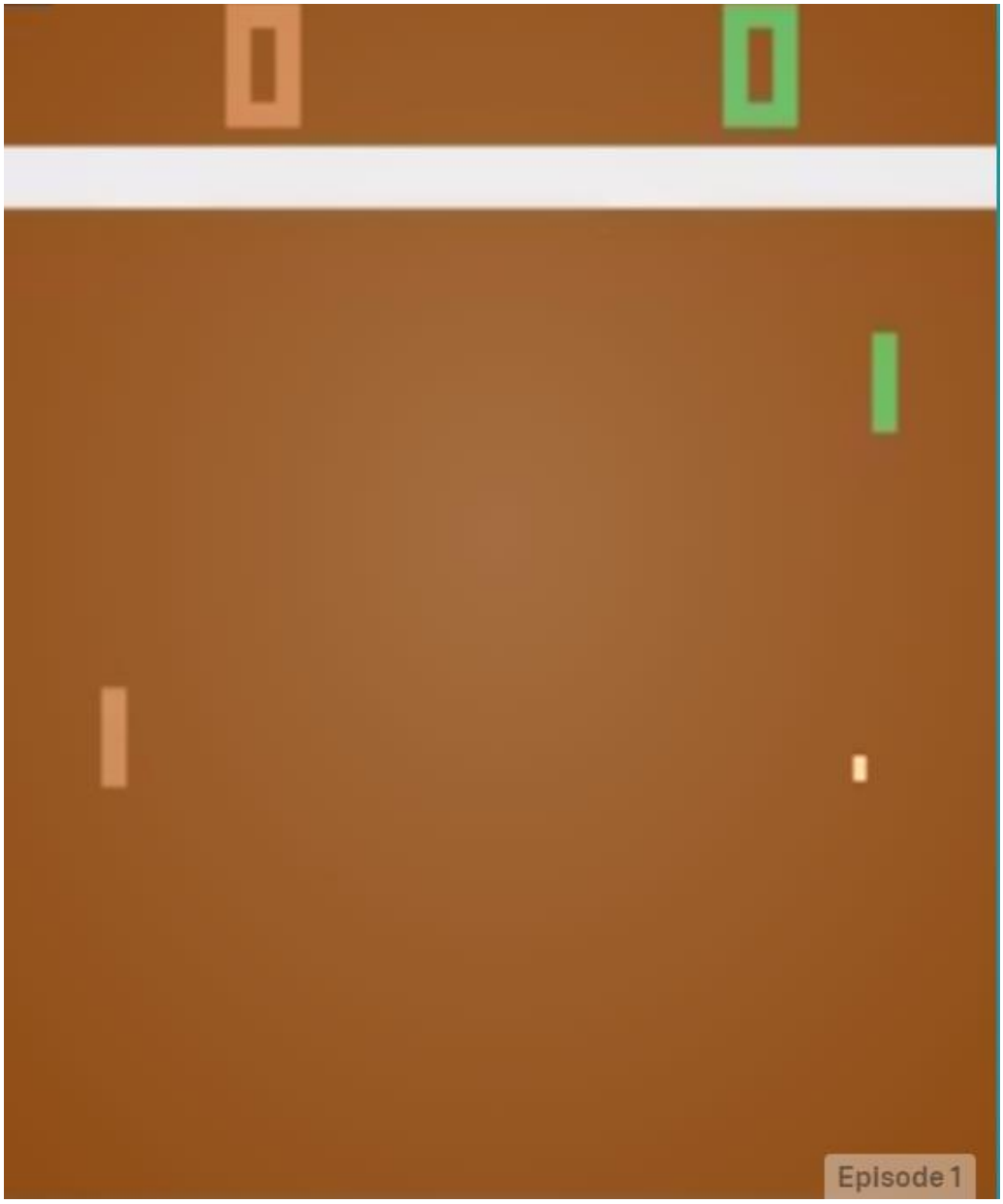


FIGURE 3 ΤΟ ΠΕΡΙΒΑΛΛΟΝ ATARI-PONG

Το δεύτερο περιβάλλον είναι αυτό του Atari Pong. Το συγκεκριμένο παιχνίδι είναι ένα παιχνίδι τύπου Ping-Pong. Ο πράκτορας κινεί την δεξιά πλατφόρμα και προσπαθεί χτυπώντας τη μπάλα να βάλει όσο περισσότερα goals στον αντίπαλό. Goal θεωρείτε όταν η μπάλα περάσει τον παίκτη και χτυπήσει το τοίχο. Στο πάνω

μέρος της οθόνης φαίνονται το Score του αντιπάλου και του παίκτη αντίστοιχα. Πιο συγκεκριμένα:

- Η παρατήρηση / είσοδος του πράκτορα για το περιβάλλον είναι τα Pixels της οθόνης μεγέθους 210x160 pixels RGB.
- Οι δυνατές κινήσεις του πράκτορα είναι 3: Καμία κίνηση, μετακίνηση του πράκτορα πάνω ή μετακίνηση του πράκτορα κάτω.
- Ο πράκτορας λαμβάνει την ανταμοιβή του στο τέλος του επεισοδίου η οποία υπολογίζεται ως το score του μείον το score του αντιπάλου. Για παράδειγμα αν έχασε 21-0 τότε υπολογίζεται ανταμοιβή -21 ενώ αν νίκησε 0-21 τότε υπολογίζεται ανταμοιβή 21 η οποία είναι και η μέγιστη που μπορεί να πάρει.
- Το επεισόδιο τερματίζει όταν ένας από τους δύο φθάσει στους 21 πόντους.
- Το επεισόδια θεωρείτε ότι έχει λυθεί αν η μέση ανταμοιβή στα τελευταία 50 επεισόδιά είναι μεγαλύτερη ή ίση του 20

Το NoFrameskip σημαίνει ότι χρησιμοποιεί το τύπο του περιβάλλοντος που δεν κάνει skip frames. Κάθε Atari περιβάλλον στο OpenAIGym επαναλαμβάνει αυτόματα στοχαστικά τη τελευταία κίνηση του πράκτορα για 2 έως 4 frames. Αυτό εισάγει στοχαστικότατα στο περιβάλλον και βοηθάει στο πράκτορα να μην απομνημονεύει απλά κινήσεις. Για κάθε περιβάλλον υπάρχει η NoFrameskip εκδοχή του που δεν πραγματοποιεί αυτή την επανάληψη. Αυτό το κάναμε γιατί βάζουμε μέσω ειδικών wrappers το Frameskip (δες επόμενο Κεφάλαιο: Wrappers)

3.2.3 MsPacmanNoFrameskip-v4

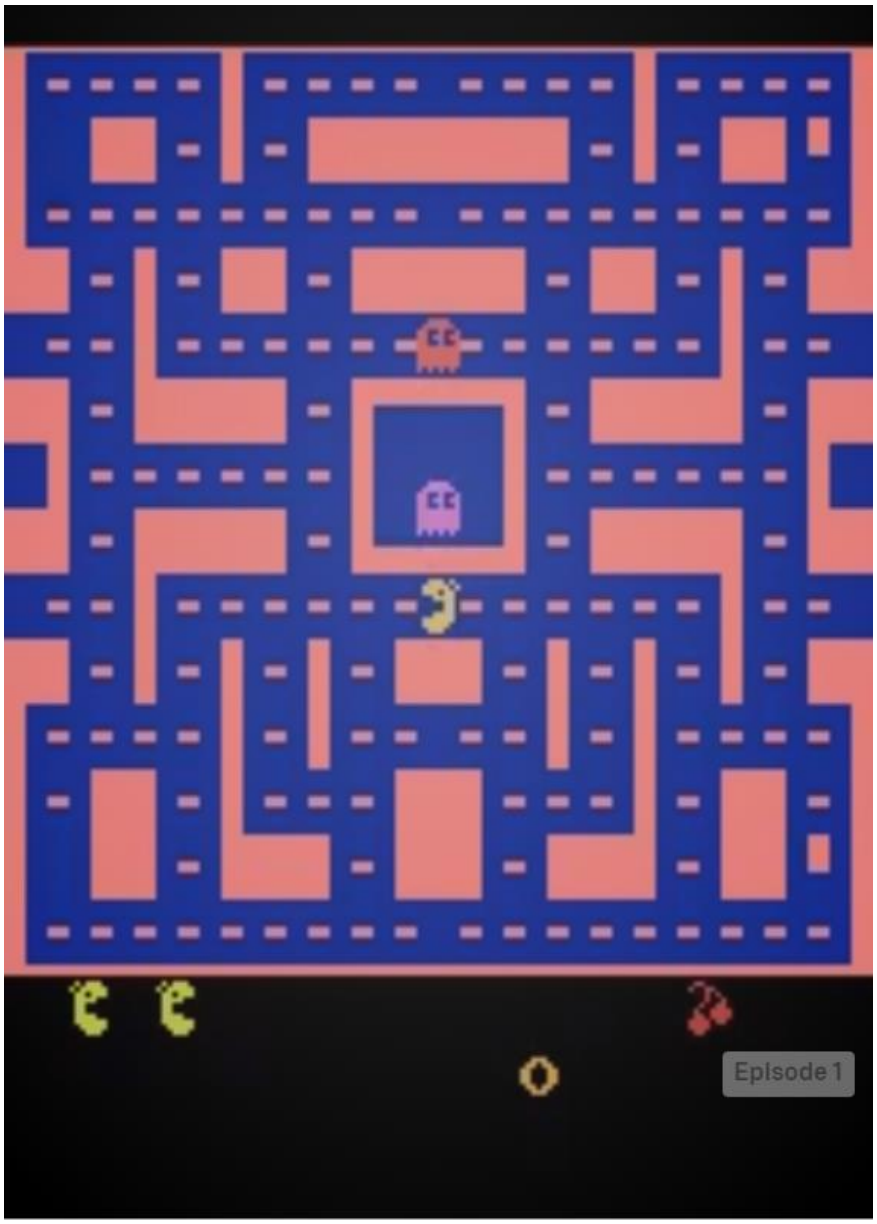


FIGURE 4 ΤΟ ΠΕΡΙΒΑΛΛΟΝ ATARI-MSPACMAN

Πρόκειται για το κλασσικό παιχνίδι MsPacman. Ο πράκτορας ελέγχει το κίτρινο τερατάκι που ξεκινάει στο κέντρο. Υπάρχουν όπως φαίνεται και στην εικόνα διάφορα φαντάσματα που κυνηγάνε τον πράκτορα και αν τον αγγίξουν χάνει μια ζωή. Ο πράκτορας ξεκινάει με 3 ζωές και πρέπει να φάει όσο περισσότερα τετραγωνάκια γίνεται. Τα μεγάλα ορθογώνια που φαίνονται στην οθόνη πάνω δεξιά και κάτω δεξιά κάνουν τον πράκτορα άτρωτο για ένα χρονικό διάστημα και επίσης αν αγγίξει σε αυτή τη φάση ένα φάντασμα τότε το τρώει για ένα χρονικό

διάστημα και παίρνει έξτρα πόντους. Αν όλα τα τετραγωνάκια είναι εξαφανισμένα το επίπεδο θεωρείτε ότι έχει τελειώσει και μεταβαίνει στο επόμενο πιο δύσκολο επίπεδο.

Στο κάτω μέρος της οθόνης φαίνονται αρχικά ο αριθμός των ζωών που έχει ο πράκτορα, οι πόντοι που έχει μαζέψει και τα κεράσια που έχει μαζέψει. Τα κεράσια εμφανίζονται σε τυχαία διαστήματα και δίνουν bonus πόντους αν τα μαζέψει. Πιο συγκεκριμένα:

- Και σε αυτή τη περίπτωση η παρατήρηση είναι μια εικόνα 210 x 160 RGB από την οθόνη
- Οι διαθέσιμες κινήσεις του πράκτορα είναι: Να μην κάνει τίποτα, να αλλάξει τη κατεύθυνση της κίνησής του προς τα δεξιά, τα αριστερά, πάνω ή κάτω
- Η ανταμοιβή που λαμβάνει είναι ίση με τα τετράγωνα που έφαγε αθροισμένη με τους μπόνους πόντους αν φάει κεράσι ή φάντασμα
- Το επεισόδιο θεωρείται ότι έχει τελειώσει αν χάσει και τις τρείς ζωές του
- Σε κάθε επανεκκίνηση ο πράκτορας ξεκινάει στη μέση του επιπέδου και όλα τα τετράγωνα επαναφέρονται
- Δεν υπάρχει κάποιος προκαθορισμένος στόχος για το πότε θεωρείτε λυμένο το πρόβλημα. Ο στόχος στη παρούσα πτυχιακή είναι να καταφέρει ο πράκτορας να ολοκληρώσει το πρώτο επίπεδο

3.2.4 Super Mario Bros 1 – Επίπεδο 1-1

FIGURE 5 ΠΕΡΙΒΑΛΛΟΝ SUPER MARIO BROS 1

Το κλασσικό παιχνίδι Super Mario Bros. Αν και το παρόν περιβάλλον δεν είναι επίσημα φτιαγμένο από το OpenAI Gym, έχει αναπτυχθεί στα πρότυπα του και παρέχει τις ίδιες λειτουργικότητες με αυτό. Το παρόν περιβάλλον περιέχει όλες τις πίστες του πρώτου Super Mario καθώς και επιλογή για τυχαία πίστα. Ο πράκτορας ελέγχει το Mario που είναι το κόκκινο ανθρωπάκι στη πάνω εικόνα. Στόχος του είναι να φθάσει στη σημαία η οποία βρίσκεται στο τέλος κάθε επιπέδου όσο πιο γρήγορα γίνεται και με τους περισσότερους πόντους. Η πόντοι που έχει μαζέψει φαίνονται στο πάνω αριστερά κομμάτι, δίπλα είναι τα

νομίσματα που έχει μαζέψει. Έπειτα, είναι ο αριθμός της πίστας και ο χρόνος που του απομένει.

Οι κλασσικοί εχθροί είναι αυτοί που φαίνονται στη πάνω εικόνα. Ο Mario μπορεί να τους σκοτώσει πηδώντας πάνω τους παίρνοντας έξτρα πόντους.

Αναλυτικότερα:

- Η είσοδος ή παρατήρηση του πράκτορα είναι μια εικόνα 240 x 256 RGB. Έχουμε να επιλέξουμε από 4 εκδόσεις ανάλογα με την ανάλυση που θέλουμε. Οι τέσσερις αυτές εκδόσεις παρουσιάζονται παρακάτω (Στην εκπαίδευση χρησιμοποιήθηκε η δεύτερη με το μαύρο φόντο):

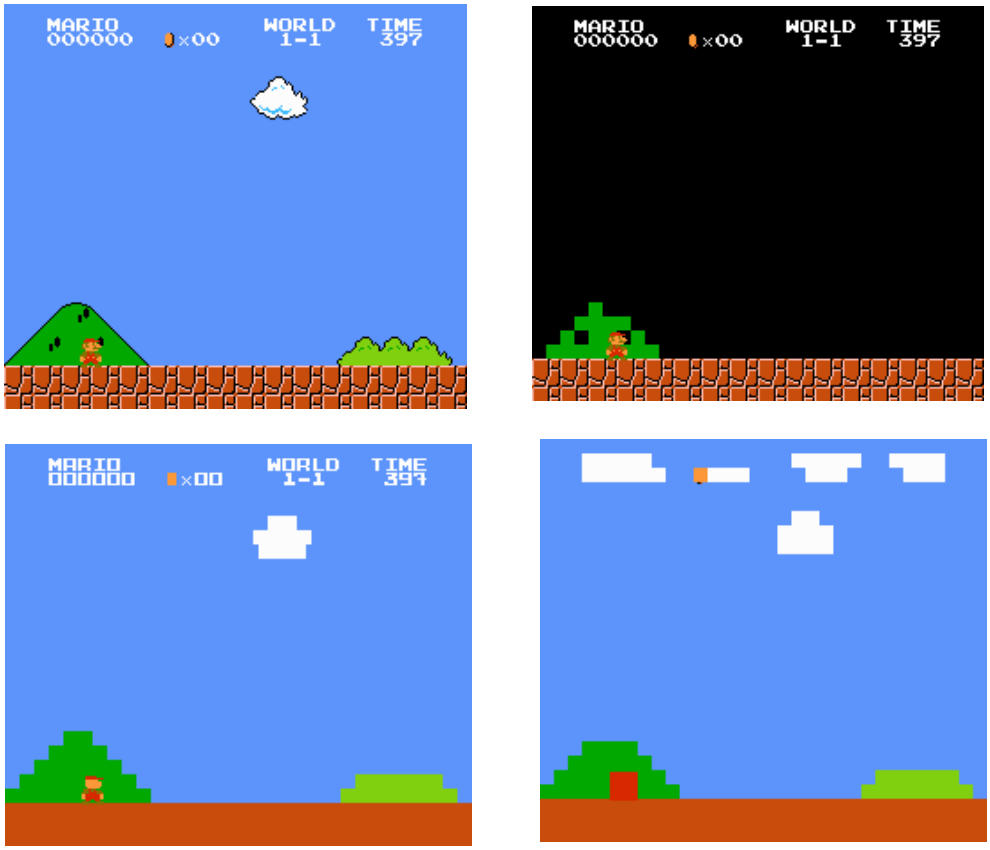


FIGURE 6 ΔΙΑΦΟΡΕΤΙΚΕΣ ΕΚΔΟΧΕΣ ΠΕΡΙΒΑΛΛΟΝΤΩΝ

- Ως κινήσεις έχουμε 3 επιλογές. Right-Only, Simple-Movement και Complex-Movement. Οι πράκτορες έχουν προπονηθεί με το Simple-Movement το οποίο περιλαμβάνει: καμία κίνηση, δεξιά, δεξιά και A (δηλαδή κίνηση

δεξιά και άλμα ταυτόχρονα), δεξιά και Β (δηλαδή κίνηση δεξιά και τρέξιμο ταυτόχρονα), δεξιά και Α και Β, Α (άλμα), αριστερά.

- Η ανταμοιβή του πράκτορα υπολογίζεται σε κάθε βήμα και είναι: r = v + c +d. Όπου v είναι η ταχύτητα του σε σχέση με το τελευταίο βήμα. Αν κινείται προς τα δεξιά η ταχύτητα είναι μεγαλύτερη του 0 ενώ προς τα αριστερά μικρότερη. C είναι η διαφορά του χρόνου πριν και μετά τη κίνηση. Αυτό προτρέπει το πράκτορα να μη μένει ακίνητος και να συνεχίσει να κινείται. Τέλος, d είναι μια ποινή ίση με -15 που δίνεται στον πράκτορα κάθε φορά που πεθαίνει. Αυτό προτρέπει το πράκτορα να αποφεύγει το θάνατο.
- Το επεισόδιο θεωρείτε ότι έχει ολοκληρωθεί αν ο πράκτορας ακουμπήσει τη σημαία που βρίσκεται στο τέλος της πίστα ή πεθάνει πριν από αυτό. Ο πράκτορας μπορεί να πεθάνει είτε από κάποιον εχθρό είτε επειδή έπεσε σε έναν γκρεμό ή τέλος επειδή του τέλειωσε ο χρόνος πριν προλάβει να φθάσει στο τέρμα της πίστας.
- Σε κάθε επανεκκίνηση ο πράκτορας ξεκινάει από την αρχή του επιπέδου με το χρόνο και το score στην αρχική τους κατάσταση.
- Δεν υπάρχει κάποια μετρική για το πότε θεωρείτε λυμένο το πρόβλημα. Στη παρούσα εργασία στόχος του πράκτορα είναι να εκπαιδευτεί να ολοκληρώνει το πρώτο επίπεδο.

3.2.5 Atari Wrappers

Και στις δυο μεθόδους μάθησης χρησιμοποιήθηκαν κάποιοί έτυμοι wrappers σχεδιασμένη για τα Atari περιβάλλοντα. Αυτοί οι wrappers επεκτείνουν τα περιβάλλοντα Atari με τις εξής λειτουργικότητες:

- EpisodicLifeEnv: Για περιβάλλοντα όπου ο πράκτορας έχει ζωές αλλά όταν χάνει μια ζωή δεν τελειώνει το παιχνίδι. Αυτός ουσιαστικά δίνει την εντύπωση στον πράκτορα ότι κάθε ζωή που χάνει είναι ένα επεισόδιο. Έτσι μαθαίνει ότι όταν πεθαίνει παίρνει λιγότερους πόντους και προσπαθεί να το αποφέυγει
- NoopResetEnv (max = 30): Αυτό σημαίνει ότι για τα πρώτα βήματα που γίνεται επανεκκίνηση του παιχνιδιού ο πράκτορας δεν πραγματοποιεί κάποια κίνηση. Ο αριθμός των βημάτων είναι στοχαστικός με μέγιστη τιμή το 30. Αυτό βοηθάει γιατί στα περισσότερα παιχνίδια υπάρχει μια καθυστέρηση κάποιων βημάτων πριν αρχίσει το παιχνίδι λόγω κάποιου Animation
- FireResetEnv: Αυτός ο Wrapper μπαίνει μόνο σε περιβάλλοντα που υπάρχει ως διαθέσιμη κίνηση η ‘Fire’. Σε αυτά τα παιχνίδια κάθε φορά που γίνεται επανεκκίνηση το περιβάλλον περιμένει τον πράκτορα να πατήσει ‘Fire’ ώστε να ξεκινήσει το παιχνίδι. Αυτό που κάνει ουσιαστικά αυτός ο wrapper είναι να εκκινεί το παιχνίδι με τη κίνηση Fire σε κάθε επανεκκίνηση
- WarpFrame: Αυτό που κάνει είναι να μειώνει την ανάλυση της αρχικής εικόνας από 210 x 160 και RGB σε 84 x 84 και ασπρόμαυρη. Αυτό μειώνει πολύ τους χρόνους εκπαίδευσης χωρίς να χάνεται κάποια σημαντική πληροφορία. Παρακάτω φαίνεται για παράδειγμα η αρχική είσοδος και η είσοδος μετά την επεξεργασία:

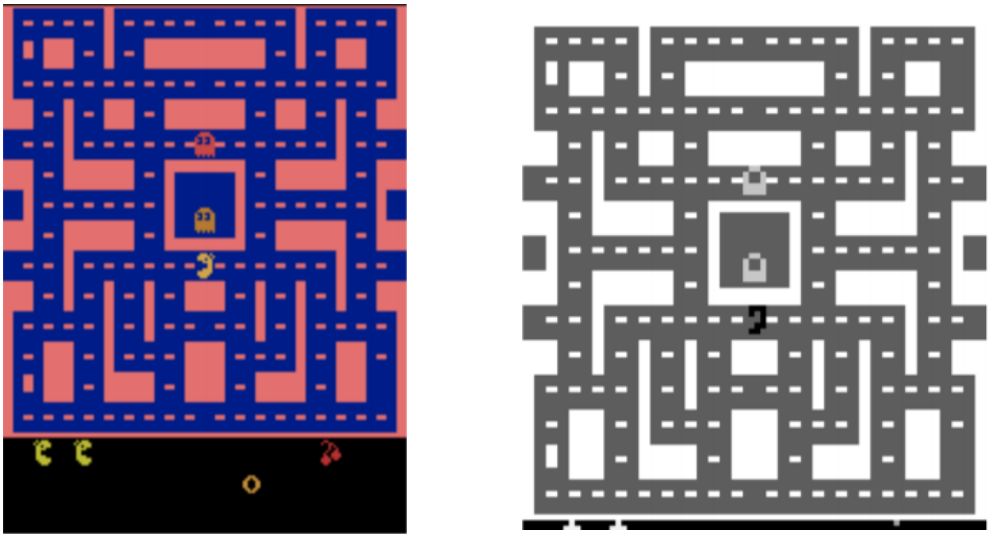


FIGURE 7 ΑΡΧΙΚΗ ΕΙΚΟΝΑ (160 X 210 RGB) ΚΑΙ ΕΙΚΟΝΑ ΜΕΤΑ ΑΠΟ ΤΗΝ ΕΠΕΞΕΡΓΑΣΙΑ (84X84X1)

- FrameStack(4): Αυτό που κάνει είναι να δίνει στον πράκτορα τα 4 τελευταία Frames του παιχνιδιού αντί για το 1 ως παρατήρηση. Αυτό είναι πολύ σημαντικό ειδικά όσον αφορά κινούμενα αντικείμενα. Ένας πράκτορας που βλέπει μόνο μια παγωμένη εικόνα για παράδειγμα δεν μπορεί να καταλάβει αν κάτι κινείται προς το μέρος του ή αν απομακρύνεται. Στο Pong για παράδειγμα αν βλέπει μόνο μια εικόνα δεν μπορεί να αναγνωρίσει αν η μπάλα κινείτε προς το μέρος του ή απομαρκύνεται. Με 4 καρέ μπορεί να αναγνωρίσει τη ταχύτητα ενός σώματος, και την επιτάχυνση.

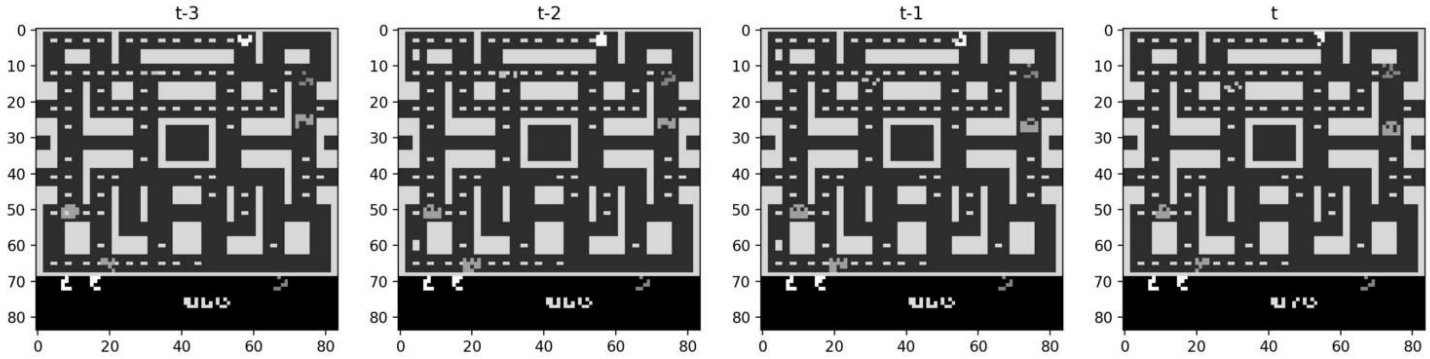


FIGURE 8: ΤΑ ΤΕΣΣΕΡΑ ΤΕΛΕΥΤΑΙΑ ΚΑΡΕ
ΣΤΟ ΠΕΡΙΒΑΛΛΟΝ MS-PACMAN

- ScaledFloatFrame: Αυτό που κάνει είναι να διαιρεί όλες τις παρατηρήσεις με το 255 για να τις φέρει στο διάστημα -1, 1. Και αυτό επίσης μειώνει πολύ τους χρόνους εκτέλεσης.
- MaxAndSkipEnv(4) : Αυτός ο wrapper διαλέγει ένα διάστημα στοχαστικά από το 2-4 κάθε φορά που πρέπει να πραγματοποιηθεί μια ενέργεια και επαναλαμβάνει την κίνηση που επέλεξε ο πράκτορας για αυτό τον αριθμό των βημάτων. Αυτό βοηθάει ώστε ο πράκτορας να μην απομνημονεύσει απλά μια σειρά κινήσεων που τον φέρνουν στο καλύτερο αποτέλεσμα, αλλά να καταλάβει καλύτερα το στόχο και να μπορεί να προσαρμόζεται σε νέες καταστάσεις.
- Clip Reward: Αυτό που κάνει είναι να φέρνει τις ανταμοιβές στο διάστημα [-1,1]. Αυτό βοηθάει στη σύγκλιση της συνάρτησης που προσπαθούμε να μεγιστοποιήσουμε / ελαχιστοποιήσουμε και στο να μην γίνουν πολύ μεγάλες οι παράγωγοι.

3.3 Βιβλιοθήκες Machine Learning

Η παρούσα πτυχιακή επίσης κάνει χρήση από έτοιμες βιβλιοθήκες μηχανικής μάθησης. Πιο συγκεκριμένα κάνει χρήση των βιλιοθηκών του Tensorflow και του Keras για τα νευρωνικά δίκτυα, του Numpy για αριθμητικές πράξεις και του Pandas για την αποθήκευση των αποτελεσμάτων και τη παρουσίαση διαγραμμάτων.

Το Tensorflow αν και στο παρελθόν ήταν ξεχωριστό από το Keras τα τελευταία χρόνια έχουν ενοποιηθεί σε μια ενιαία πλατφόρμα όπου μέσω του Keras δημιουργούνται τα διάφορα νευρωνικά δίκτυα ή μοντέλα που θα χρησιμοποιηθούν και μέσω του tensorflow η υπόλοιπη λειτουργικότητα.

Οι βιβλιοθήκες Keras-Tensorflow παρέχουν:

- Εύκολη δημιουργία νευρονικών δικτύων μέσω έτοιμων τύπων κρυφών επιπέδων, συναρτήσεων ενεργοποίησης κλπ.
- Εύκολη εκπαίδευση των δικτύων μέσω διαφόρων έτοιμων μεθόδων εκπαίδευσης,
- Υλοποιημένες συναρτήσεις σφάλματος, διαχείριση ρυθμού μάθησης, μέθοδοι αρχικοποίησεις βαρών
- Επιπλέον λειτουργικότητα όπως μεταβλητός ρυθμός μάθησης, σημεία ελέγχου, loggin
- Δυνατότητα επέκτασης των παραπάνω με συγγραφή κώδικα από το χρήστη

Τέλος, ο κώδικας είναι υλοποιημένος σε python καθώς έχει το μεγαλύτερο εύρος από έτοιμες βιβλιοθήκες μηχανικής μάθησης.

4 Μέθοδος Προσέγγισης Συνάρτησης Αξιών (Function Value Approximation)

4.1 Εισαγωγικά

Η παρουσίαση της μεθόδου θα γίνει ως εξής: Αρχικά, θα πραγματοποιηθεί μια ανάλυση του αλγορίθμου και το πώς λειτουργεί. Έπειτα, θα παρουσιαστούν τα αποτελέσματα των πειραμάτων που έγιναν σε διαφορετικά περιβάλλοντα με διαφορετικούς στόχους και θα σχολιαστούν. Τέλος, θα προβάλω τα συμπεράσματα μου, τις δυσκολίες που αντιμετώπισε ο αλγόριθμος και διάφορες προτάσεις για βελτίωση.

4.2 Περιγραφή Αλγορίθμου: Deep Q Learning

4.2.1 Προσέγγιση Συνάρτηση Ωφέλειας και Νευρωνικά Δίκτυα

Στις προσεγγιστικές μεθόδους οι πράκτορες προσπαθούν να ανακαλύψουν τη συνάρτηση μέγιστης ωφέλειας ενέργειας $q_*(s,a)$ για κάθε ζεύγος δυνατών (s,a). Ο λόγος που αναζητάει το $q_*(s,a)$ και όχι το $v_*(s)$ είναι ότι από μόνο του το $v_*(s)$ δεν πληροφορεί τον πράκτορα για το ποια είναι η καλύτερη ενέργεια που μπορεί να ακολουθήσει παρά μόνο πόσο καλή είναι η κατάσταση που βρίσκεται.

Εν αντίθεσί, όπως προαναφέρθηκε και στην ενότητα 2, αν ο πράκτορας γνώριζε τη συνάρτηση μέγιστης ωφέλειας $q_*(s,a)$ ενέργειας για κάθε ζεύγος καταστάσεων (s, a) τότε τα πράγματα θα ήταν απλά. Θα επέλεγε σε κάθε κατάσταση την ενέργεια που του δίνει το μεγαλύτερο q_* και έτσι θα συμπεριφερόταν με το βέλτιστο δυνατό τρόπο. Εδώ προκύπτουν δυο προβλήματα: Που αναπαρίστανται η συνάρτηση μέγιστης ωφέλειας ενέργειας για κάθε ζεύγος (s,a) και πώς μπορεί να την ανακαλύψει χωρίς τη γνώση των δυναμικών του περιβάλλοντος.

Θα ξεκινήσω με το πρώτο, το πως αναπαρίστανται η πληροφορία για το $q_*(s,a)$. Κάθε ζεύγος κατάστασης-ενέργειας (s,a) έχει μια διαφορετική τιμή q_* ανάλογα τον τύπο του περιβάλλοντος και τους στόχους που πρέπει να πραγματοποιήσει σε αυτό. Αν το πρόβλημα ήταν αρκετά μικρό με λίγες καταστάσεις και ενέργειες τότε αυτή η πληροφορία θα μπορούσε να αποθηκευτεί σε ένα δισδιάστατο πίνακα όπου κάθε γραμμή αντιστοιχεί σε μια κατάσταση και κάθε στήλη σε μια

ενέργεια. Σε μια τέτοια περίπτωση για να βρει τη συνάρτηση μέγιστης χρησιμότητας ενέργειας αρκεί να κοιτάξει στην αντίστοιχη γραμμή και στήλη.

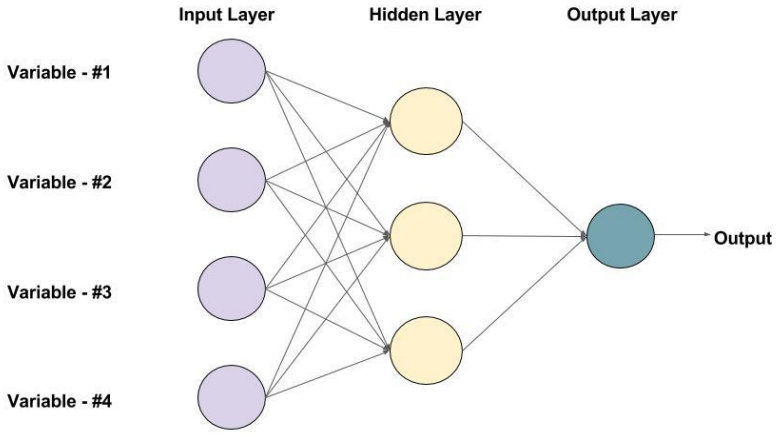
Σε μεγάλα προβλήματα όμως, όπως αυτά που ερευνώ στη συγκεκριμένη πτυχιακή ή προβλήματα του φυσικού κόσμου κάτι τέτοιο είναι αποτρεπτικό. Το πλήθος των δυνατών καταστάσεων είναι τεράστιου όγκου και οι ενέργειες πολλές ή μερικές φορές συνεχείς.

Η λύση που έχει προταθεί σε αυτό το πρόβλημα είναι η προσέγγιση της συνάρτησης χρησιμότητας. Δηλαδή η κατασκευή ενός μηχανισμού που δέχεται τα χαρακτηριστικά μιας κατάστασης ως είσοδο και βγάζει μια προσέγγιση για το q_*. Έτσι έχουμε εξαλείψει την ανάγκη για αποθήκευση του q_* για κάθε ζεύγους (s,a) και ο στόχος γίνεται η προσέγγιση αυτών μέσω μιας συνάρτησης.

Η προσέγγιση της συνάρτησης αυτής μπορεί να γίνει με έναν από τους παρακάτω τρόπους:

- Γραμμικός Συνδυασμός των χαρακτηριστικών / εισόδου
- Νευρωνικό Δίκτυο
- Δέντρο Απόφασης
- Κοντινότεροι Γείτονες

Στη παρούσα πτυχιακή γίνεται χρήση της δεύτερης μεθόδου δηλαδή ενός νευρωνικού δικτύου για τη προσέγγιση της συνάρτησης q_*. Τα Νευρωνικά Δίκτυα ή αλλιώς Τεχνητά Νευρωνικά Δίκτυα ξεκίνησαν ως μια προσπάθεια μοντελοποίησης του ανθρώπινου εγκεφάλου. Αποτελούνται από διάφορους κόμβους, οι αποκαλούμενοι ΄Νευρώνες’, και διάφορα επίπεδα. Οι νευρώνες μπορεί να είναι συνδεμένοι μεταξύ τους με διάφορους τρόπους η πιο κλασσική αρχιτεκτονική Νευρωνικού Δικτύου όμως είναι αυτή του πλήρως συνδεδεμένου προς τα μπρος δικτύου (feed-forward neural network) όπως φαίνεται στην εικόνα:



An example of a Feed-forward Neural Network with one hidden layer (with 3 neurons)

Όπως φαίνεται στη παραπάνω εικόνα το δίκτυο αποτελείται από τρία επίπεδα. Οι νευρώνες του κάθε επιπέδου συνδέονται με τους νευρώνες του επόμενου επιπέδου. Κάθε ακμή που συνδέει νευρώνες έχει ένα βάρος w_{ij}. Το πρώτο επίπεδο ονομάζεται και επίπεδο εισόδου και έχει μέγεθος όσες οι είσοδοι στο δίκτυο, ενώ το τελευταίο ως επίπεδο εξόδου και έχει μέγεθος όσες οι έξοδοι που θέλουμε να παράγει. Όλα τα μεσαία επίπεδα χαρακτηρίζονται ως 'κρυφά' επίπεδα.

Κάθε νευρώνας εκτός από αυτούς στην είσοδο παράγει την έξοδο του ως εξής: Αρχικά παίρνει το γραμμικό συνδυασμό των νευρώνων του προηγούμενου επιπέδου με τα αντίστοιχα βάρη που τους συνδέουν. Αν για παράδειγμα συμβολίσουμε με $a_i^{(\kappa)}$ την έξοδο του νευρώνα i στο κ επίπεδο τότε η έξοδος του νευρώνα j στο (κ+1) επίπεδο υπολογίζεται ως: $a_j^{k+1} = \sum_{i=1}^{n} w_{ij} * a_i^{\kappa}$.

Συνήθως αυτή η έξοδος περνάει από μια συνάρτηση ενεργοποίησης. Διάφορες δημοφιλείς συναρτήσεις ενεργοποίησης είναι η σιγμοειδής, η Relu, Elu, βηματική κλπ.

Τα Νευρωνικά δίκτυα με την προσθήκη διαφόρων κρυφών επιπέδων έχουν καταφέρει να προσεγγίσουν διάφορες περίπλοκες συναρτήσεις. Έχουν προταθεί διάφορες συναρτήσεις κόστους ή σφάλματος που μετρούν πόσο απέχει η συνάρτηση που έχει προσεγγίσει το νευρωνικό δίκτυο από τη ζητούμενη. Ο σκοπός στα νευρωνικά δίκτυα είναι να βρεθούν οι παράμετροι w που δίνουν τα αναμενόμενα αποτελέσματα δηλαδή μοντελοποιούν μια συνάρτηση και ελαχιστοποιούν τη συνάρτηση κόστους. Ο τρόπος με τον οποίο εκπαιδεύονται τα νευρωνικά δίκτυα για να βρεθού αυτές οι παράμετροι w είναι γνωστός ως 'Κανόνας Δέλτα' και βασίζεται στον υπολογισμού του ανάδελτα ή διανύσματος κλίσεων της συνάρτησης κόστους ως προς κάθε παράμετρο w. Όπως είναι γνωστό από τα μαθηματικά το διάνυσμα κλίσεων της συνάρτησης κόστους ως προς κάθε w μας δίνει τη κατεύθυνση της πιο απότομης ανάβασης δηλαδή της κατεύθυνσης που μεγιστοποιεί πιο γρήγορα το κόστος από το συγκεκριμένο σημείο. Επομένως για να την ελαχιστοποιήσουμε αλλάζουμε τα βάρη προς την αντίθετη κατεύθυνση και έτσι προκύπτει η αλλαγή των βαρών μετά από κάθε επανάληψη του αλγορίθμου:

$$\Delta wij = -a * \nabla \mathrm{C} \longrightarrow$$

Όπου C η συνάρτηση κόστους και α ο ρυθμός μάθησης. Ο ρυθμός μάθησης επιλέγεται ξεχωριστά για κάθε πρόβλημα και δείχνει το πόσο απότομη είναι η αλλαγή των βαρών σε κάθε βήμα του αλγορίθμου. Πρέπει να τον επιλέξουμε με προσοχή γιατί μια πολύ μικρή τιμή του α οδηγεί σε πολύ αργή σύγκλιση ενώ με μια πολύ μεγάλη τιμή μπορεί να αποκλίνει συνεχώς από το ολικό ελάχιστο και να μη συγκλίνει ποτέ. Ο τρόπος να βρούμε το α είναι με πολλές δοκιμές και σύγκριση.

Η συνάρτηση που θέλουμε σε αυτή τη περίπτωση να προσεγγίσει το δίκτυο είναι αυτή της μέγιστης ωφέλειας ενέργειας q_*. Εδώ μπορούμε να επιλέξουμε μεταξύ δυο προσεγγίσεων και οι δυο είναι αποδεκτές: Η πρώτη είναι το δίκτυο να παίρνει τα χαρακτηριστικά μιας κατάστασης s και μια ενέργεια ως είσοδο και να βγάζει την προσέγγιση της $q_*(s,a)$ ή να παίρνει μόνο τα χαρακτηριστικά της κατάστασης και να βγάζει για έξοδο τη προσέγγιση $q_*(s,a)$ για κάθε δυνατή ενέργεια από τη συγκεκριμένη κατάσταση.

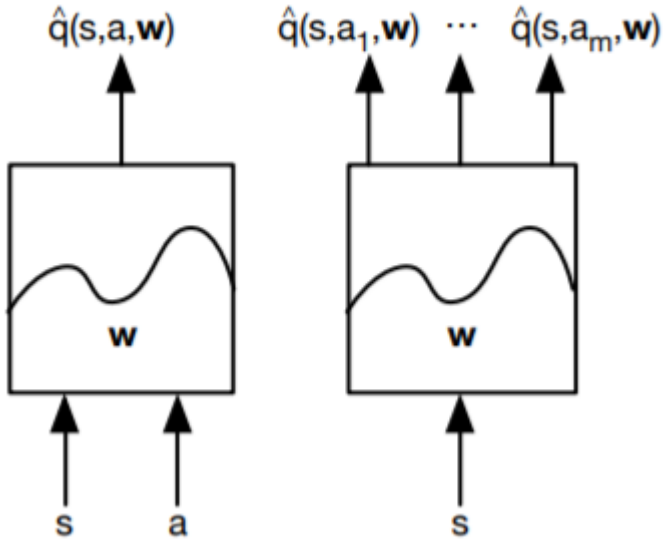


FIGURE 10 ΔΥΟ ΔΙΑΘΕΣΙΜΕΣ ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ ΤΟΥ ΝΕΥΡΩΝΙΚΟΥ ΔΙΚΤΥΟΥ

Για τους σκοπούς της εργασίας και χάριν ευκολίας έχουμε επιλέξει τη δεύτερη μέθοδο.

Η αρχιτεκτονική του πλήρως συνδεδεμένου προς τα εμπρός δικτύου δεν είναι η μοναδική αλλά μέσα στα χρόνια έχουν παρουσιαστεί διάφορες αρχιτεκτονικές

δικτύων που εξειδικεύονται σε διαφορετικές εργασίες. Ο τύπος του δικτύου που χρησιμοποιείται στη παρούσα πτυχιακή είναι τα λεγόμενα ‘Συνελεκτικά Δίκτυα’ (Convolution Networks).

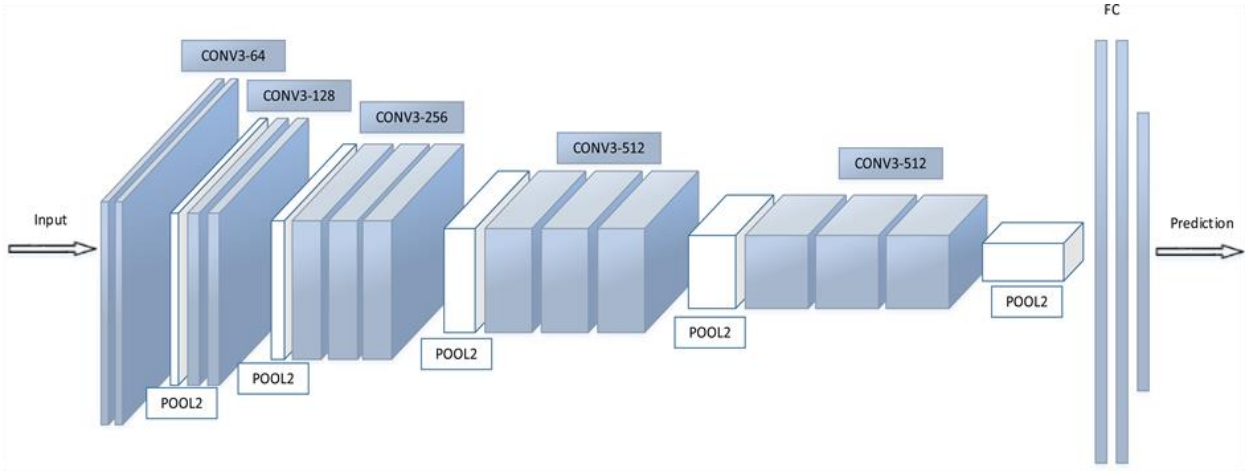


FIGURE 11 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΝΕΛΕΚΤΙΚΟΥ ΔΙΚΤΥΟΥ

Αυτά τα δίκτυα όπως φαίνεται στο παραπάνω σχήμα έχουν ένα δισδιάστατο πίνακα για είσοδο και δεν τον στειροποιούν όπως τα κλασσικά προς τα εμπρός δίκτυα. Είναι ειδικά σχεδιασμένα για την αναγνώριση αντικειμένων σε εικόνες και εκεί έχουν βρει τεράστια επιτυχία. Ο τρόπος που λειτουργούν είναι ο εξής: Αρχικά δέχονται μια δισδιάστατη εικόνα με βάθος 1 ή μεγαλύτερο. Το βάθος για παράδειγμα μπορεί να είναι τα Red Green και Blue επίπεδα στο RGB. Τα επόμενα επίπεδα έχουν όπως ονομάζονται διάφορους πυρήνες. Αυτοί οι πυρήνες έχουν ένα εύρος ή σκοπιά που αναλύουν κάθε φορά διάφορες περιοχές της εικόνας και προσπαθούν να εξάγουν διαφορετικά χαρακτηριστικά. Έτσι, παράγουν τους διάφορες χάρτες χαρακτηριστικών όπως ονομάζονται οι οποίοι αναλύονται στα επόμενα επίπεδα από τους επόμενους πυρήνες. Αυτού η διαδικασία συνεχίζεται μέχρι το τελευταία επίπεδα. Τα τελευταία επίπεδο συνήθως στειροποιεί τους χάρτες χαρακτηριστικών και αποτελείται από κάποια πλήρως συνδεδεμένα επίπεδα που τελικά παράγουν την έξοδο/εξόδους.

Στη περίπτωση που μας ενδιαφέρει η είσοδος του δικτύου θα είναι τα 4 τελευταία καρέ της οθόνης, και η έξοδος που θα περιμένουμε θα έχει μέγεθος όσο και οι διαθέσιμες ενέργειες και θα εκφράζει τη συνάρτηση χρησιμότητας από τη κατάσταση s για κάθε ενέργεια a.

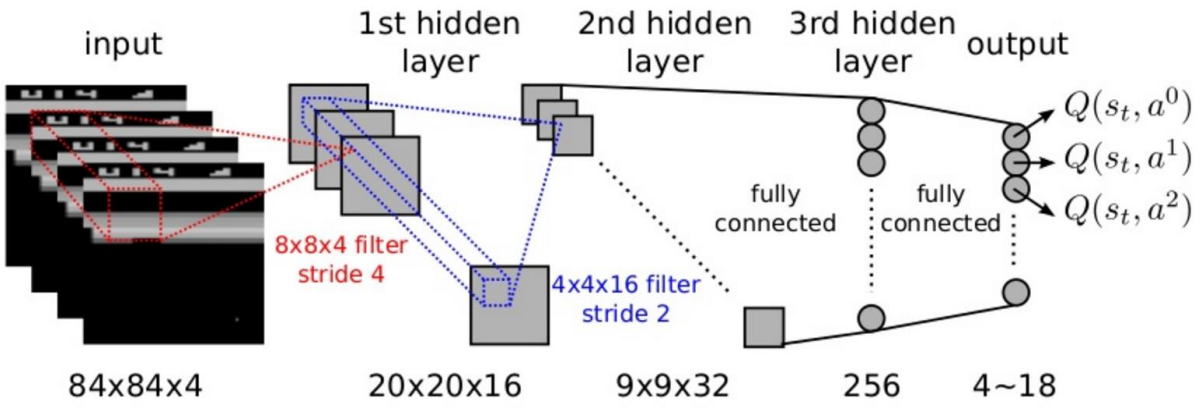


FIGURE 12 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΔΙΚΤΥΟΥ ΓΙΑ ATARI ΠΕΡΙΒΑΛΛΟΝΤΑ

Η αρχιτεκτονική του δικτύου που χρησιμοποιείται στη παρούσα πτυχιακή είναι λίγο διαφορετική από αυτή της παραπάνω εικόνας. Η είσοδος ισούται όπως την εικόνα με τα 4 τελευταία καρέ της οθόνης και η έξοδος με το πλήθος των δυνατών ενεργειών. Η διαφοροποίηση γίνεται στα κρυφά επίπεδα, όπου η υλοποίηση μου στο πρώτο κρυφό επίπεδο περιλαμβάνει 32 πυρήνες μήκους 8x8 με stride = 4. Το stride δείχνει το πόσο μεγάλα βήματα κάνει μέσα στην εικόνα. Για μεγάλους πυρήνες μπορεί να είναι μεγάλο γιατί καλύπτουν μεγάλες περιοχές τις εικόνας. Στο 2ο κρυφό επίπεδο υπάρχουν 64 4x4 πυρήνες με stride=2. Το 3ο κρυφό επίπεδο περιλαμβάνει 64 πυρήνες 3x3 με stride=1. Τέλος, μετά που σειροιοποιηθούν οι έξοδοι των πυρήνων περνάνε σε ένα πλήρως συνδεδεμένο επίπεδο με 512 νευρώνες. Όλοι οι νευρώνες έχουν για συνάρτηση ενεργοποίησης τη Relu εκτός από αυτούς της εξόδους που έχουν τη γραμμική.

Επίσης, μια μικρή διαφοροποίηση που χρησιμοποιήθηκε στο δίκτυο της εργασίας είναι η προσθήκη μιας μάσκας στην είσοδο, με μέγεθος όσο το πλήθος όλων των εισόδων στο περιβάλλον και η παραγωγή της εξόδου με πολλαπλασιασμό πινάκων των q_* της εξόδου με τη μάσκα. Αυτό θα μας παρέχει μια διευκόλυνση όταν περάσουμε στο ζήτημα της εκπαίδευσης.

Ως συνάρτηση κόστους χρησιμοποιήθηκε μια παραλλαγή της συνάρτησης μέσου τετραγωνικού σφάλματος. Πρόκειται για μια ενδιάμεση συνάρτηση μεταξύ αυτής του μέσου τετραγωνικού σφάλματος και του απόλυτου σφάλματος η οποία έχει αποδειχθεί ότι είναι πιο ανεθεκτική σε εξωτερικές τιμές (outliers). Ο ορισμός της συνάρτησης είναι:

$$\frac{1}{2} * \left(y - f(x)\right)^2 \ \gamma\iota\alpha \ |y - f(x)| \leq \delta$$

$$\delta * |y - f(x)| - \frac{1}{2} * \delta^2 \ \gamma\iota\alpha \ |y - f(x)| > \delta$$

Αυτό που λένε η παραπάνω δυο συναρτήσεις είναι γοα σφάλμα μικρότερο από δ χρήση του μέσου τετραγωνικού σφάλματος ενώ για μεγαλύτερο από δ χρήση μέσου απόλυτου σφάλματος. Η επιλογή μιας τιμής του δ έχει μεγάλη σημασία. Μετά από διάφορα τεστ η καλύτερη τιμή που βρέθηκε ήταν το 2.

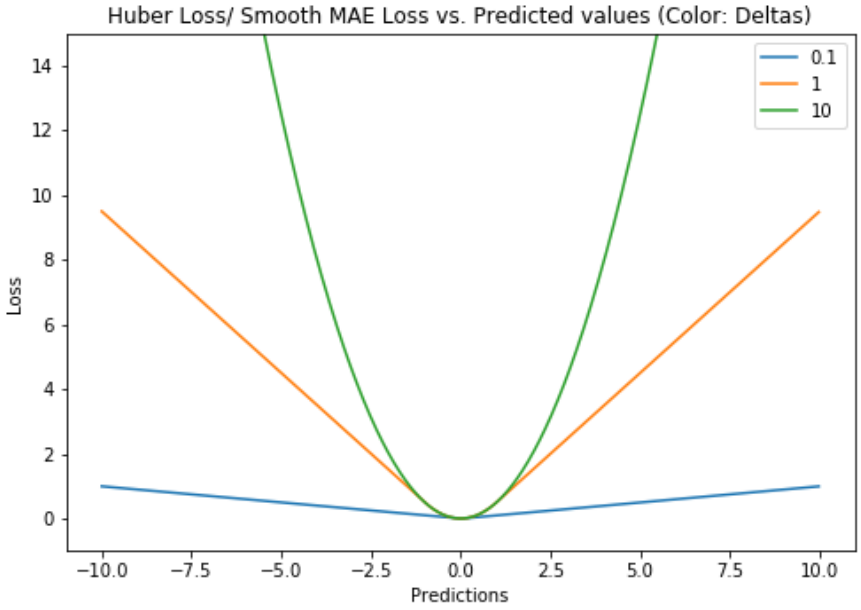


FIGURE 13 Η ΣΥΝΑΡΤΗΣΗ ΣΦΑΛΜΑΤΟΣ HUBBER (HUBBER LOSS)

Τέλος, για την ελαχιστοποίηση χρησιμοποιήθηκε η μέθοδος Adam. Η συγκεκριμένη μέθοδος βασίζεται σε αυτή του κανόνα δέλτα αλλά περιλαμβάνει στους υπολογισμούς του τις προηγούμενες κλίσεις και τα τετράγωνα τον προηγούμενων κλίσεων με ορμή που τον βοηθάνε να συγκλίνει πιο γρήγορα στη βέλτιστη λύση.

4.2.2 Εκπαίδευση Δικτύου μέσω Q-Learning

Τώρα περνάμε στο δεύτερο πρόβλημα που είναι το πώς θα προσεγγίσουμε τη συνάρτηση χρησιμότητας ενέργειας q_*. Αν είχαμε τις πραγματικές τιμές q_* στα

χέρια μας τότε η εκπαίδευση του δικτύου θα γινόταν με το κλασσικό τρόπο της επιβλεπόμενης μάθησης. Στις ενισχυτικές μεθόδους όμως δεν γνωρίζουμε εξαρχής τη πραγματική τιμή q_* και πρέπει να την ανακαλύψουμε. Πως λοιπόν μπορούμε να εφαρμόσουμε το κανόνα δέλτα για να προσεγγίσουμε τη συνάρτηση μέγιστης ωφέλειας ενώ δεν γνωρίζουμε τις πραγματικές τιμές;

Η λύση σε αυτό το πρόβλημα έγκειται στις επαναλαμβανόμενες εμπειρίες που έχει ο πράκτορας και στη συνάρτηση ανταμοιβής. Αναλυτικότερα, ως η πραγματική τιμή της συνάρτησης μέγιστης ωφέλειας $q_*(s,a)$ χρησιμοποιείται η ανταμοιβή που έλαβε ο πράκτορας μετά από τη πραγματοποίηση της ενέργειας R_t αθροισμένη με τη προεξόφληση της εκτίμησης της συνάρτησης ανταμοιβής του δικτύου για την επόμενη κατάσταση S_{t+1} αν πραγματοποιήσει την ενέργεια a_{t+1}. Έπειτα μπορεί να χρησιμοποιηθεί ο κανόνας δέλτα της επιβλεπόμενης μάθησης για να ανανεωθούν τα βάρη:

$$\Delta w = a * \left(R_t + \gamma * \hat{q}(S_{t+1}, a_t, w) - \hat{q}(S_t, a_t, w)\right) * \nabla_w \hat{q}(S_t, a_t, w)$$

Όπου αυτό που είναι σημειωμένο με μπλε είναι η 'πραγματική' τιμή της συνάρτησης μέγιστης ωφέλειας. Αυτό θα προσεγγίζει όλο και περισσότερο στη πραγματική τιμή μέγιστης ωφέλειας όσο ο αριθμός των επαναλήψεων αυξάνεται.

Έτσι βρέθηκε ένας τρόπος να προσεγγιστεί η πραγματική τιμή της συνάρτησης μέγιστης ωφέλειας χωρίς τη γνώση της πραγματικής τιμής της. Το μόνο που χρειάζεται για την ενημέρωση των βαρών σε μια κατάσταση S_t μετά από μια κίνηση a_t είναι η ανταμοιβή που έλαβε ο πράκτορας, η εκτίμηση της συνάρτησης ωφέλειας για την κατάσταση 'S_t' και ενέργεια 'α_t', και η εκτίμηση της συνάρτησης ωφέλειας στην επόμενη κατάσταση S_{t+1} με κίνηση a_{t+1}. Όλα τα παραπάνω μπορούν να βρεθούνε με μια ενέργεια του πράκτορα ενώ αλληλοεπιδρά με το περιβάλλον.

Σε αυτό το σημείο είναι χρήσιμο να γίνει ο διαχωρισμός μεταξύ δυο μεθόδων προσέγγισης της συνάρτησης μέγιστης ωφέλειας ενέργειας. Αυτών των On-Policy (πάνω στη στρατηγική) και αυτών των Off-Policy (εκτός στρατηγική).

Οι On-Policy αλγόριθμοι προσπαθούν να μάθουν τη πολιτική/στρατηγική 'π' ακολουθώντας αυτή τη πολιτική 'π' και παίρνοντας εμπειρία από αυτή. Κάποιοι αλγόριθμοι αυτής της κατηγορίας είναι οι:

- Monte-Carlo
- Sarsa – TD Learning
- TD(λ)

Οι αλγόριθμοι αυτής της κατηγορίας είναι έξω από τη σκοπιά της παρούσας πτυχιακής και δεν θα αναλυθούν.

Εν αντίθεσί οι Off-Policy αλγόριθμοι μαθαίνουν για μια πολιτική ‘π’ ενώ ακολουθούν και λαμβάνουν εμπειρίες από μια άλλη πολιτική ‘μ’. Για παράδειγμα μια πολύ συχνή επιλογή των π και μ είναι η μάθηση μιας καθαρά άπληστης στρατηγικής ‘π’, ενώ ακολουθεί μια άπληστη στρατηγική με πιθανότητα ε. Αυτή είναι η βασική ιδέα του Q-Learning, ενός Off-Policy αλγορίθμου που χρησιμοποιείται στη παρούσα πτυχιακή.

Αναλυτικότερα, ο πράκτορας σε όλα τα σημεία της εκπαίδευσης ακολουθεί μια ε-άπληστη στρατηγική (ε-greedy). Αυτή η στρατηγική λέει στον πράκτορα να επιλέξει την ενέργεια η οποία έχει τη μεγαλύτερη στη παρούσα φάση εκτίμηση q(s,a) για τη κατάσταση που βρίσκεται, με πιθανότητα μεγαλύτερη του ε, διαφορετικά να επιλέξει μια τυχαία ενέργεια με ίση πιθανότητα. Η παράμετρος ε παίζει μεγάλο ρόλο στο θέμα της αναζήτησης έναντι εκμετάλλευσης που σχολιάστηκε στην ενότητα 2.

Μια μεγάλη τιμή της παραμέτρου ε, ωθεί τον πράκτορα να επιλέξει τυχαίες κινήσεις και άρα να αναζητήσει για νέα μονοπάτια, ενώ μια μικρή τιμή του ε τον ωθεί να επιλέξει τη καλύτερη για αυτόν κίνηση. Αυτό που έκανα σε αυτή τη πτυχιακή για να πετύχω μια ισορροπία εκμετάλλευσης / εξερεύνησης είναι στην αρχή του αλγορίθμου που η συνάρτηση μέγιστης χρησιμότητας ενέργειας δεν είναι γνωστή και έχει τυχαία συμπεριφορά το ε είναι ίσο με 1 δηλαδή παίρνει όλες τις κινήσεις τυχαία και εξερευνάει το χώρο και σε κάθε βήμα του αλγορίθμου μειώνεται το ε μέχρι μια τελική τιμή που είναι 0.01 ή 0.05 ή και 0.1. Οι παράμετροι του πλήθους του αριθμού των βημάτων μέχρι να πέσει το ε στη κατώτερη τιμή του, δηλαδή το πόσο γρήγορα μειώνεται ή η τελική τιμή του ε εξαρτάται κάθε φορά από το περιβάλλον. Για παράδειγμα ένα πιο απλοϊκό περιβάλλον μπορεί να μη χρειάζεται τόσο εξερεύνηση και για αυτό η παράμετρος ε να πέφτει σε πιο λίγα βήματα στη τελική της τιμή.

Μια καθαρά άπληστη στρατηγική είναι ουσιαστικά η επιλογή πάντα της ενέργειας που δίνει το μεγαλύτερο q με πιθανότητα 1 (ε=0). Επομένως πρόκειται για μια στρατηγική πλήρους εκμετάλλευσης.

Έτσι στον Q Learning αλγόριθμο, ο πράκτορας ακολουθεί μια ε-άπληστη στρατηγική και προσπαθεί να τη βελτιώσει σύμφωνα με την άπληστη πολιτική. Τα παραπάνω μπορούν να εκφραστούν με μαθηματικό τρόπο ως εξής:

$$\Delta w = a * \left(R_t + \gamma * \max_{a'} \hat{q}(S_{t+1}, a', w) - \hat{q}(S_t, a_t, w) \right) * \nabla_w \hat{q}(S_t, a_t, w)$$

Και πάλι με μπλε σημειώνεται η 'πραγματική' τιμή που θα έπρεπε να βγάλει το δίκτυο αν πραγματοποιούσαμε επιβλεπόμενη μάθηση. Η διαφορά σε σχέση με το προηγούμενο τύπο είναι στο ότι ο υπολογισμός της συνάρτησης χρησιμότητας για την επόμενη κατάσταση γίνεται σε σχέση με την ενέργεια που δίνει τη μεγαλύτερη χρησιμότητα. Δεν είναι απαραίτητο ότι ο πράκτορας θα την ακολουθήσει στο επόμενο βήμα. Αντιθέτως, στο προηγούμενο τύπο ο υπολογισμός της συνάρτησης χρησιμότητας στην επόμενη κατάσταση γινόταν με βάση τη στρατηγική που ακολουθούσε ο πράκτορας για παράδειγμα ε-άπληστη και μετά ακολουθούσε αυτή την ενέργεια. Φαίνεται λοιπόν η διαφοροποίηση από των On-policy και των Off-Policy μεθόδων.

Με βάση τον παραπάνω τύπο γίνονται οι ανανεώσεις των βαρών στα βήματα του αλγορίθμου. Η συνάρτηση κόστους που αλγόριθμος προσπαθεί να ελαχιστοποιήσει είναι ως εξής:

$$L(w) = E[\left(R_t + \gamma * \max_{a'} \hat{q}(s_{t+1}, a'w) - \hat{q}(s_t, a_t, w) \right)^2]$$

Είναι ο κλασσικός τύπος του μέσου τετραγωνικού σφάλματος και μπορεί εύκολα να μετασχηματιστεί στη συνάρτηση σφάλματος Hubber.

Αν και σε πιο απλά προβλήματα αυτή η ενημέρωση των βαρών θα είναι αρκετή για να βρεθεί η βέλτιστη συμπεριφορά, σε πιο σύνθετα προβλήματα όπως αυτά που εξετάζονται δεν θα καταφέρει να προσεγγίσει το ολικό ελάχιστο. Για τη καταπολέμηση αυτού έχουν προταθεί κάποιες βελτιώσεις που κάνουν τον αλγόριθμο πιο λειτουργικό. Η πρώτη είναι η χρήση μιας μνήμης από που αντλεί

τις εμπειρίες του ο πράκτορας και το δεύτερο είναι η σταθεροποίηση των βαρών. Στην επόμενη ενότητα παρουσιάζονται αναλυτικότερα και τα δύο.

4.2.3 Βελτιώσεις στον Q-Learning

Η πρώτη βελτίωση στη λειτουργικότητα του αλγορίθμου συνδέεται με την ακολουθία των εμπειριών που δέχεται ο πράκτορας. Αν ο πράκτορας εκπαιδεύεται σε ένα περιβάλλον δέχεται κάποιες εμπειρίες σε μια ακολουθιακή σειρά. Η συνεχής επανάληψη και η εκπαίδευση σε αυτές τις εμπειρίες έχει ως αποτέλεσμα ο πράκτορας να δημιουργεί εξαρτήσεις μεταξύ των παρατηρήσεων που δέχεται και να υπερεκπαιδεύεται (overfit) στις εμπειρίες που δέχεται. Ακόμη, είναι πιο αποτελεσματικό να εκπαιδεύεται σε ομάδες δεδομένων (batches) παρά στο καθένα ξεχωριστά.

Η πρώτη βελτίωση που έχει προταθεί για τον αλγόριθμο καταπολεμά τα παραπάνω δυο προβλήματα. Αυτό το πετυχαίνει με τη διατήρηση μιας μνήμης στην μεριά του πράκτορα, στην υλοποίηση μου μήκος 1 εκατομμυρίου καταχωρήσεων. Κάθε καταχώριση περιλαμβάνει τη κατάσταση στην οποία βρισκόταν (δηλαδή τα 4 τελευταία στιγμιότυπα οθόνης) την ενέργεια που πραγματοποιείσαι, την ανταμοιβή που έλαβε από αυτήν και την επόμενη κατάσταση στην οποία βρέθηκε. Ο πράκτορας εκπαιδεύεται αποθηκεύοντας κάθε εμπειρία που δέχεται από το περιβάλλον στη μνήμη του και μετέπειτα αντλώντας κάθε 4 βήματα μια ομάδα ανακατεμένων παραδειγμάτων μήκους 32 από τη μνήμη του και η ενημέρωση των βαρών του δικτύου με βάση αυτά.

Μια παραπέρα βελτίωση που έχει προταθεί είναι αντί τα παραδείγματα εκπαίδευσης να αντλούνται από τη μνήμη με τυχαίο τρόπο, να γίνεται η εισαγωγή τους στη μνήμη με βάση μια προτεραιότητα και η επιλογή τους με βάση αυτή. Η προτεραιότητα αυτή ορίζεται ως το απόλυτο σφάλμα δηλαδή η διαφορά της πραγματικής τιμής με αυτή της τιμής που προέβλεψε ο πράκτορας. Αυτό έχει ως αποτέλεσμα ο πράκτορας να εκπαιδεύεται με μεγαλύτερη πιθανότητα στα παραδείγματα που οι εκτιμήσεις του αποκλίνουν περισσότερο από τη πραγματική τιμή και με μικρότερη σε αυτά που έχει εκπαιδευτεί αρκετά ή πλήρως. Αυτό επιταχύνει πολύ το χρόνο σύγκλισης και βοηθάει το πράκτορα στο να εκπαιδεύεται περισσότερο στα παραδείγματα που χρειάζεται περισσότερη βελτίωση.

Η υλοποίηση της μνήμης προτεραιότητας μπορεί να γίνει με διάφορους τρόπους. Ένας τρόπος είναι η διάταξη των παραδειγμάτων στη μνήμη με σειρά προτεραιότητας. Αυτό έχει πολύ χαμηλή απόδοση και θα προϋπέθετε τη συνεχή διάταξη των παραδειγμάτων κάθε φορά που εισέρχεται ένα νέο στη μνήμη. Έχει προταθεί μια πιο αποδοτική υλοποίηση από το σύνδεσμο που φαίνεται στο παρακάτω σύνδεσμο και την οποία χρησιμοποιώ στη παρούσα εργασία: https://jaromiru.com/2016/11/07/lets-make-a-dqn-double-learning-and-prioritized-experience-replay/ .

Αναλυτικότερα, η παραπάνω υλοποίηση χρησιμοποιεί ένα δένδρο αθροίσματος ως τη δομή δεδομένων της μνήμης. Το δένδρο αθροίσματος είναι ένα δυαδικό δένδρο στο οποίο κάθε γονιός αποθηκεύει ως τιμή το άθροισμα των παιδιών του. Οι πραγματικές καταχωρήσεις αποθηκεύονται στους κόμβους παιδιά του δένδρου. Στο παράδειγμά μας οι κόμβοι παιδία αποθηκεύουν τις εμπειρίες του πράκτορα μαζί με τις αντίστοιχες προτεραιότητές τους ενώ οι γονείς περιέχουν το άθροισμα των προτεραιοτήτων των παιδιών τους. Η ρίζα περιλαμβάνει το άθροισμα όλων των προτεραιοτήτων του δένδρου.

Έτσι, αν θέλουμε να πάρουμε τυχαία μια καταχώρηση με βάση τη προτεραιότητα αρχικά επιλέγουμε έναν τυχαίο αριθμό από το 0 μέχρι τη τιμή που έχει ρίζα. Έπειτα διασχίζουμε το δένδρο σύμφωνά με το παρακάτω απλό ψευδοκώδικα μέχρι να φτάσουμε σε ένα κόμβο παιδί.

ΨΕΥΔΩΚΩΔΙΚΑΣ ΑΝΑΚΤΗΣΗ_ΕΓΓΡΑΦΗΣ(ΚΟΜΒΟΣ, ΤΥΧΑΙΟΣ_ΑΡΙΘΜΟΣ):

```
Αν Κομβος = Κόμβος_παιδί:
    Επιστροφή Κόμβος_παιδί
Αν Αρ_Κομβος >= Τυχαιος_Αριθμός:
    Ανακτηση_Εγγραφης(Αριστερός_κόμβος, Τυχαίος_αριθμός)
Αλλιώς
    Ανακτηση_Εγγραφης(Δεξιός_κόμβος,
    Τυχαίος_αριθμός – Αριστερός_κόμβος)
```

Η υλοποίηση αυτή μοιάζει στη φύση της στη συνάρτηση σωρευτικής πιθανότητας. Έτσι αρκεί να υπολογιστεί το σφάλμα κάθε εμπειρίας πριν προστεθεί στη μνήμη και ο αλγόριθμος θα την επιλέξει με μεγαλύτερη ή μικρότερη προτεραιότητα ανάλογα με το απόλυτο σφάλμα της. Τέλος, πρέπει να ενημερώσουμε τη προτεραιότητα κάθε καταχώρησης κάθε φορά που επιλέγεται για εκπαίδευση ώστε να συμπίπτει με το τρέχον δίκτυο.

Περνάμε ύστερα στο επόμενο πρόβλημα που έχει παρατηρηθεί στην αρχική μέθοδο εκπαίδευσης. Έχει παρατηρηθεί ότι η συνεχής ενημέρωση των βαρών σε πολύ τακτά χρονικά διαστήματα κάνει το δίκτυο ασταθές και το δυσκολεύει στη σύγκλιση. Η λύση που έχει προταθεί είναι η χρήση ενός δεύτερου δικτύου το οποίο έχει πιο παλιά βάρη και με βάση αυτό ο υπολογισμός της πραγματικής συνάρτησης ωφέλειας. Το δίκτυο αυτό δεν εκπαιδεύεται και τα βάρη του ενημερώνονται με αυτά του κύριου δικτύου κάθε 10 χιλιάδες βήματα. Έτσι η συνάρτηση σφάλματος μετατρέπεται σε:

$$L(w_{new}) = E[\left(R_t + \gamma * \max_{a'} \hat{q}(s_{t+1}, a' w_{old}) - \hat{q}(s_t, a_t, w_{new})\right)^2]$$

Όπου σύμφωνα με τα προηγούμενα τα R_t, a_t, s_t και s_{t+1} είναι ομάδες παραδειγμάτων μήκους 32 από τη μνήμη. Έτσι λαμβάνοντας όλα τα παραπάνω υπόψη ο τελικός αλγόριθμος που προκύπτει είναι:

Αλγόριθμος Q:

env.make()

Κατασκευή και Αρχικοποίηση των δυο δικτύων με τυχαία βάρη (localnetwork, target_network)

Αρχικοποίηση Μνήμης

Για αριθμό επαναλήψεων:

Obs = env.reset()

done = False

Όσο δεν έχει τελειώσει το επισόδειο: (done != True)

// Επιλογή Ενέργειας σύμφωνα με παρατήρηση και local network

action = e-greedy(local_network, Obs, ρυθμός εξερεύνησης)

// Πραγματοποίηση ενέργειας, επιστροφή νέας παρατήρησης, ανταμοιβής και σημαίας για το αν τελείωσε το επισόδειο

next_obs, reward, done = env.step(action)

Υπολογισμός προτεραιότητας p= {

abs(reward – local_network(obs) αν done = True

*abs(reward + ρυθμός_προεξόφλησης * max(target_network(obs)) – local_network(obs)*

}

Εισαγωγή(obs, action, reward, done, next_obs) στη μνήμη με προτεραιότητα p

Αν είναι ώρα για προπόνηση:

Ομάδα_παραδειγμάτων = sample(memory, batch_size) (batch_obs, batch_reward, batch_action, batch_next_obs, batch_done)

Q_target = max(target_network(batch_next_obs))

Q_target[done] = 0.0

*Πραγματικές_τιμές = batch_reward + ρυθμός προεξόφλησης * Q_target*

Ενημέρωση προτεραιοτήτων ομάδας παραδειγμάτων

Μάσκα = One_hot(action_batch)

Σφάλμα = Local_network.train_on_batch([batch_obs, Μάσκα], target=Πραγματικές_τιμές)

Αν είναι ώρα για ενημέρωση του target_network:

target_network.set_weights(local_network.get_weights())

Παρατηρήσεις:

(1): Όπως φαίνεται παραπάνω σε περίπτωση που έχει τελειώσει το επεισόδιο δεν λαμβάνεται υπόψη στον υπολογισμό της πραγματικής τιμής η max q τιμής της επόμενης παρατήρησης και υπολογίζεται απλά ως η τρέχον ανταμοιβή

(2): Η χρήση μάσκας γίνεται για να ληφθούν υπόψη μόνο εκείνες οι τιμές q (έξοδοι του δικτύου) που αντιστοιχούν στις αντίστοιχες ενέργειες. Η μάσκα έχει τη μορφή ενός one hot διανύσματος δηλαδή έχει μήκος όσο οι διαθέσιμες ενέργειες και περιέχει μηδενικά σε όλες τις θέσεις εκτός από τη θέση της ενέργειας στην οποία έχει 1. Η μάσκα αυτή πολλαπλασιάζεται με τις τιμές q για να δώσει την έξοδο του δικτύου με αποτέλεσμα να παράγει έξοδο 0 σε όλες τις υπόλοιπες q τιμές. Έτσι το δίκτυο προπονείται με βάση μόνο την ενέργεια που πραγματοποιείσαι και την αντίστοιχη q(s,a). Σε περίπτωση που θέλουμε να πάρουμε όλες τις τιμές q γίνεται χρήση μιας μάσκας με άσσους σε όλες τις θέσεις.

Έτσι αν το δίκτυο εκπαιδευτεί για ένα μεγάλο αριθμό βημάτων (μεγαλύτερο των 30 εκατομμυρίων) θα προσεγγίσει τελικά τη μέγιστη συνάρτηση χρησιμότητας q_*. Ύστερα από τη προπόνηση του δικτύου ο πράκτορας συμπεριφέρεται με τη καλύτερη συμπεριφορά, δηλαδή αυτή που του μεγιστοποιεί τη συνολική ανταμοιβή αν από κάθε παρατήρηση / κατάσταση s επιλέγει την έξοδο του δικτύου με τη μεγαλύτερη τιμή q_* και πραγματοποιεί την αντίστοιχη ενέργεια.

Στην επόμενη ενότητα παρουσιάζονται τα αποτελέσματα του παραπάνω δικτύου μαζί με τις αντίστοιχες παραμέτρους που χρησιμοποιήθηκαν για την εκπαίδευση του αλγορίθμου.

4.3 Αποτελέσματα Εκπαίδευσης

4.3.1 CartPole-v0

Αυτό το απλοϊκό περιβάλλον χρησιμοποιήθηκε αρχικά για την εξακρίβωση της λειτουργικότητας του αλγορίθμου. Επειδή οι είσοδοι του δικτύου είναι 4 διανύσματα και όχι στιγμιότυπο της οθόνης έγινε χρήση ενός μόνο πλήρως συνδεδεμένου κρυφού επιπέδου με 24 νευρώνες.

Παράμετροι:

Ρυθμός Προεξόφλησης: 0.99

Ρυθμός Μάθησης: 0.0004

Ρυθμός Εξερεύνησης : Αρχική τιμή 1 (τυχαία) - > Τελική Τιμή 0.01 μέσα σε 20000 βήματα

Συχνότητα Προπόνησης: 4 βήματα

Συχνότητα Ενημέρωσης Target_network: 2000 βήματα

Μέγεθος Μνήμης: 300000

Η ανταμοιβή του πράκτορα μέσα στα πρώτα 4000 βήματα φαίνεται παρακάτω:

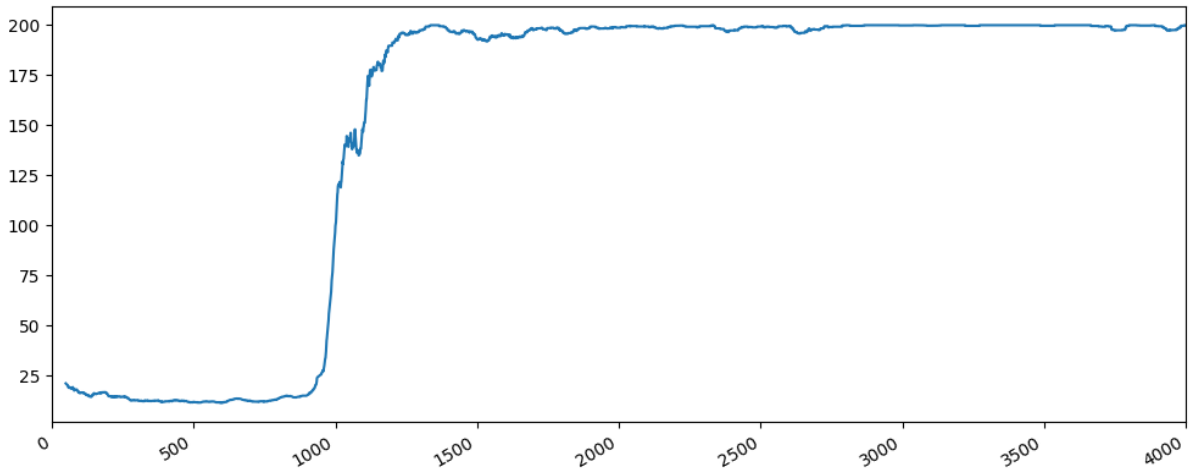


FIGURE 14 ΑΝΤΑΜΟΙΒΕΣ ΣΤΗ ΦΑΣΗ ΤΗΣ ΕΚΠΑΙΔΕΥΣΗΣ DEEP Q– ΠΕΡΙΒΑΛΛΟΝ CARTPOLE)

Όπως φαίνεται από το παραπάνω διάγραμμα ο πράκτορας μετά από 1500 βήματα κατάφερε να μεγιστοποιήσει την ανταμοιβή του στη μέγιστη (200) με μικρές διακυμάνσεις.

4.3.2 Atari: PongNoFrameskip-v4

Περνάμε τώρα στα Atari περιβάλλοντα. Σε όλα τα Atari περιβάλλοντα τα δίκτυα που χρησιμοποιούνται είναι συνελικτικά με την αρχιτεκτονική που περιγράφτηκε στην ενότητα 4.2.1. Οι παράμετροι επίσης είναι κοινές και είναι αυτές που φαίνονται παρακάτω:

Παράμετροι Atari:

Ρυθμός Προεξόφλησης: 0.99

Ρυθμός Μάθησης: 0.000025

Ρυθμός Εξερεύνησης : Αρχική τιμή 1 (τυχαία) - > Τελική Τιμή 0.01 μέσα σε 400000 βήματα

Συχνότητα Προπόνησης: 4 βήματα

Συχνότητα Ενημέρωσης Target_network: 10000 βήματα

Μέγεθος Μνήμης: 1 εκατομμύριο εγγραφές

Παρακάτω φαίνονται οι ανταμοιβές στη φάση της προπόνησης με τις παραπάνω παραμέτρους για 4000 επεισόδιά (περίπου 30 εκατομμύρια βήματα):

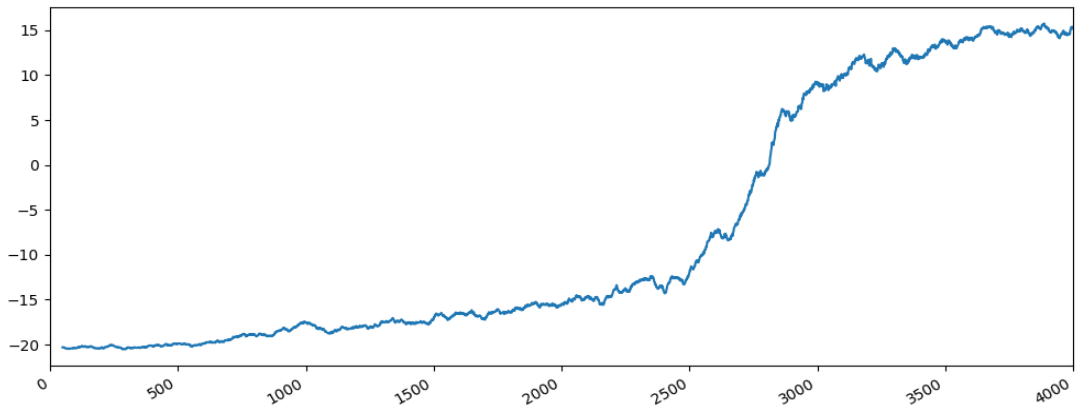


FIGURE 15 ΑΝΤΑΜΟΙΒΕΣ ΣΤΗ ΦΑΣΗ ΤΗΣ ΕΚΠΑΙΔΕΥΣΗΣ DEEP Q - ΠΕΡΙΒΑΛΛΟΝ ATARI PONG

Όπως μπορούμε να δούμε οι ανταμοιβές συνεχίζουν να αυξάνουν μέχρι το τέλος της προπόνησης και φθάνουν ως 16-17 που σημαίνει ότι οι πόντοι που σκόραρε ο πράκτοράς έχουν διαφορά 16-17 από τους πόντους του αντιπάλου.

Οι ανταμοιβές του πράκτορα στη φάση του testing για 100 επεισόδιά φαίνονται παρακάτω:

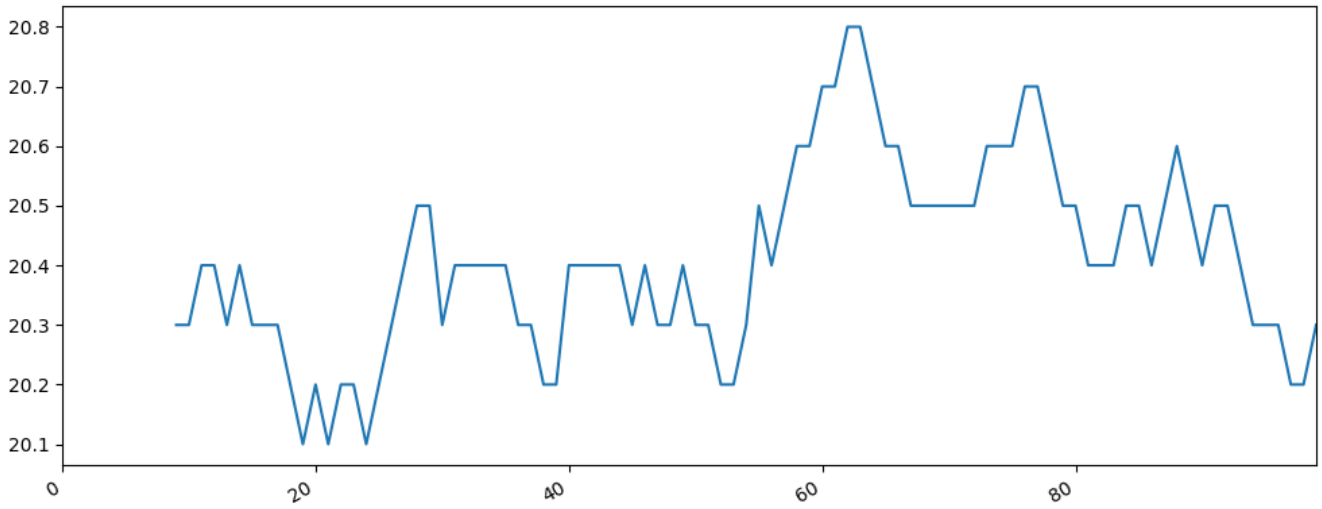


FIGURE 16 ΑΝΤΑΜΟΙΒΕΣ ΣΤΗ ΦΑΣΗ TESTING DEEP Q - ΠΕΡΙΒΑΛΛΟΝ ATARI PONG

Όπως φαίνεται οι ανταμοιβές του πράκτορα κυμαίνονται από το 20 μέχρι το 20.8 που σημαίνει ότι ο πράκτορας έχει μάθε τη βέλτιστη για αυτόν στρατηγική.

4.3.3 Super Mario Bros Πίστα 1-1

Οι ανταμοιβές του πράκτορα για το περιβάλλον Super Mario Bros στη φάση της εκπαίδευσης για 18 χιλιάδες επεισόδιά φαίνονται παρακάτω:

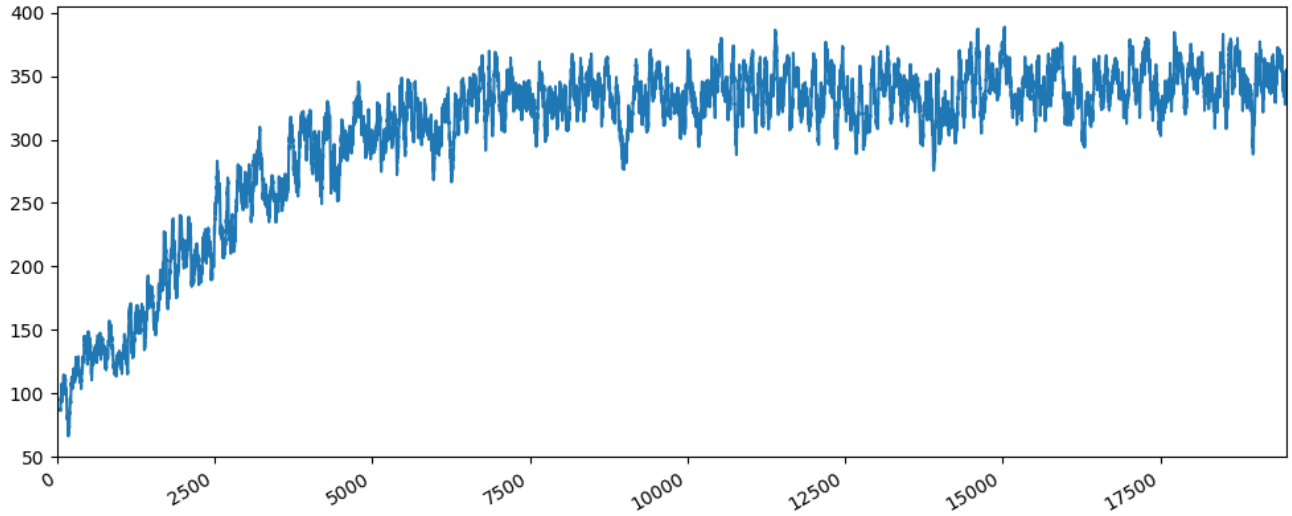


FIGURE 17 ΑΝΤΑΜΟΙΒΕΣ ΣΤΗ ΦΑΣΗ ΤΗΣ ΕΚΠΑΙΔΕΥΣΗΣ DEEP Q - ΠΕΡΙΒΑΛΛΟΝ SUPER MARIO 1-1

Όπως φαίνεται και από αυτό το διάγραμμα οι ανταμοιβές ξεκίνησαν από το 50-100 και μέχρι το τέλος έφθασαν στο εύρος 350-400 που ο πράκτορας τερματίζει το επίπεδο.

Τέλος, παρακάτω φαίνονται τα αποτελέσματα στη φάση του testing για τον πράκτορα με τη βέλτιστη στρατηγική σε 100 επεισόδια:

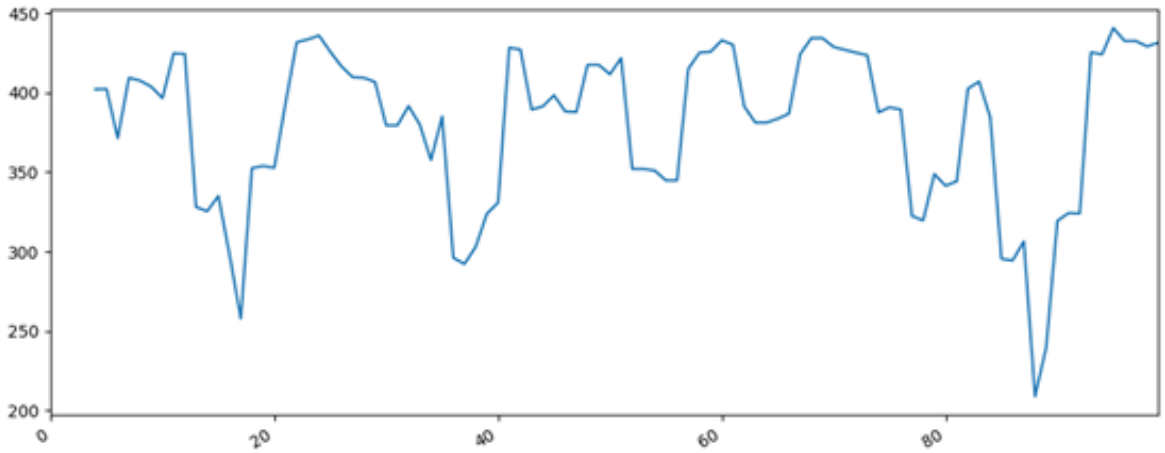


FIGURE 18 ΑΝΤΑΜΟΙΒΕΣ ΣΤΗ ΦΑΣΗ TESTING DEEP Q- ΠΕΡΙΒΑΛΛΟΝ SUPER MARIO 1-1

Όπως φαίνεται η ανταμοιβή κυμαίνεται από το 200 μέχρι το 450 ενώ η κύρια τιμή της είναι στο 400. Φαίνεται λοιπόν ότι ο πράκτορας έχει μάθει να τερματίζει τη πίστα στις περισσότερες προσπάθειες και μάλιστα με καλό σκορ (χρόνος + πόντοι)

4.3.4 Atari - MsPacman

Τέλος, παρακάτω φαίνονται οι ανταμοιβές στο MsPacman για 100 επεισόδια εκπαίδευσης:

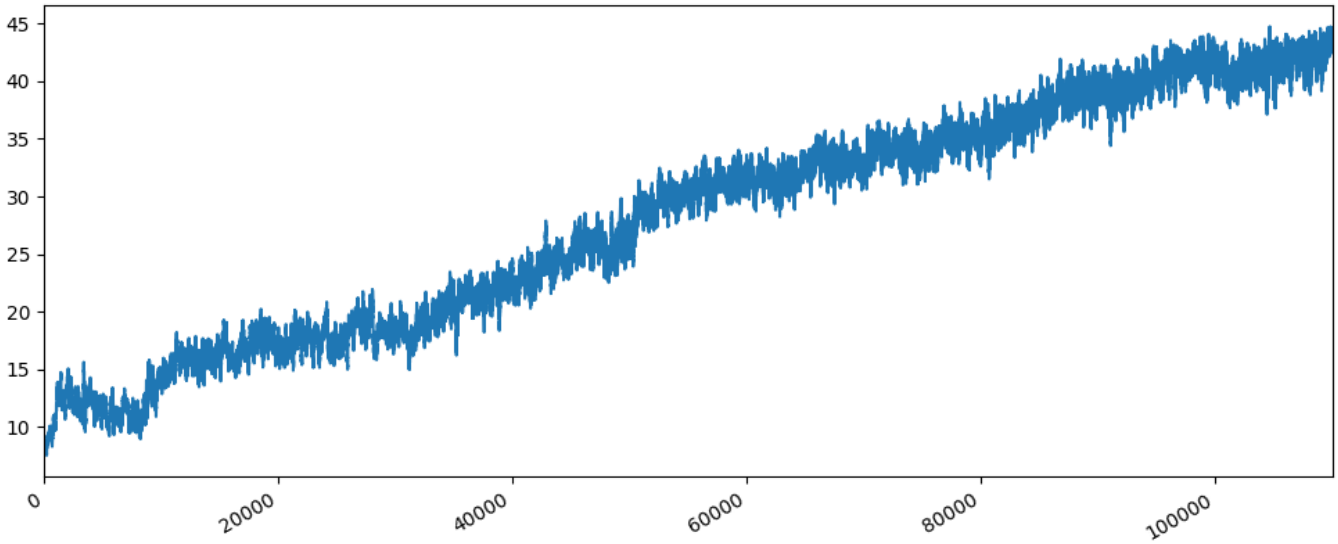


FIGURE 19 ΑΝΤΑΜΟΙΒΕΣ ΣΤΗ ΦΑΣΗ ΤΗΣ ΕΚΠΑΙΔΕΥΣΗΣ DEEP Q- ΠΕΡΙΒΑΛΛΟΝ ATARI-MSPACMAN

Και εδώ οι ανταμοιβές φαίνονται να έχουν αυξητική τάση, με αρχική τιμή το 0-10 και τελική το 45. Στο τέλος της εκπαίδευσης ο πράκτορας φθάνει πολύ κοντά στο να τελειώσει το πρώτο επίπεδο.

Παρακάτω φαίνονται οι ανταμοιβές του για 100 επεισόδια στη φάση του testing:

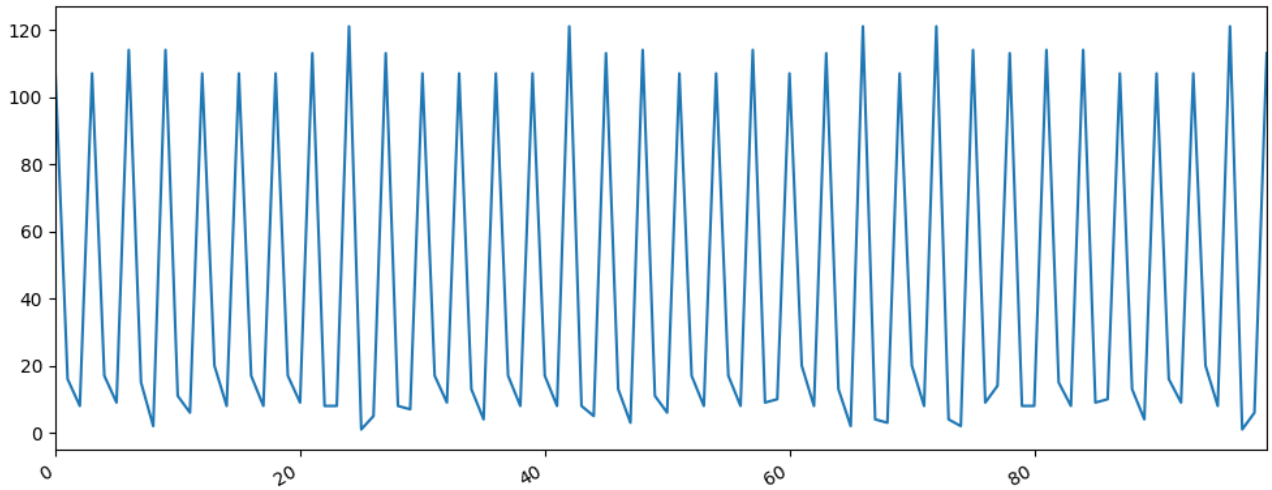


FIGURE 20 ΑΝΤΑΜΟΙΒΕΣ ΣΤΗ ΦΑΣΗ TESTING DEEP Q- ΠΕΡΙΒΑΛΛΟΝ ATARI-MSPACMAN

Ο λόγος που παρατηρούνται αυτές οι έντονες αυξομειώσεις είναι λόγω του EpisodicEnv Wrapper (δες ενότητα Wrappers). Επειδή κάθε ζωή θεωρείται σαν ένα ξεχωριστό επεισόδιο ενώ το επεισόδιο δεν τερματίζει ο πράκτορας μπορεί να πάρει πιο εύκολα μεγάλο αριθμό από πόντους στο πρώτο επεισόδιο που δεν έχει φαγωθεί καμιά κουκίδα, ενώ στις επόμενες δύο ζωές είναι δυσκολότερο γιατί υπάρχουν λιγότερες κουκίδες στο επίπεδο άρα και πόντοι. Έτσι όπως φαίνεται στο παραπάνω διάγραμμα ο πράκτορας παίρνει ανταμοιβή της τάξης του 100 έως του 120 στο πρώτο επεισόδιο που το επίπεδο είναι γεμάτο κουκίδες και στα επόμενα παίρνει χαμηλή ανταμοιβή (20-50).

Τέλος, ακόμη μια δυσκολία στο συγκεκριμένο περιβάλλον που πρέπει να ανακαλύψει ο πράκτορας είναι το πότε είναι καλό και πότε κακό να πέσει πάνω στους εχθρούς. Αν για παράδειγμα έχει φάει μόλις μια μεγάλη κουκίδα που τον κάνει άτρωτο και για τα επόμενα λίγα δευτερόλεπτα τότε είναι καλή κίνηση το να πέσει πάνω στους εχθρούς γιατί τους εξουδετερώνει και κερδίζει μπόνους πόντους. Σε όλες τις άλλες περιπτώσεις είναι κακό γιατί χάνει μια ζωή και άρα τερματίζει το επεισόδιο. Από τα παραπάνω φαίνεται ότι χρειαζόταν μεγαλύτερος αριθμός επαναλήψεων καθώς στο σημείο που διακόπηκε ακόμη βελτιωνόταν.

4.4 Συμπεράσματα DQN – Προτάσεις για Βελτίωση

Από τα παραπάνω αποτελέσματα φαίνεται ότι οι ανταμοιβές του πράκτορα παρουσιάζουν μια συνεχόμενη αυξητική πορεία και άρα όντως εκπαιδεύεται στους στόχους του κάθε περιβάλλοντος επιτυχώς. Πρέπει να δοθεί ιδιαίτερη προσοχή στη παράμετρο του ρυθμού μάθησης καθώς μια μικρή τιμή της κάνει την εκπαίδευση πολύ αργή ενώ μια μεγάλη έχει ως αποτέλεσμα ο πράκτορας να μη προσεγγίζει τη συνάρτηση μέγιστης αναμενόμενης χρησιμότητας ενέργειας.

Η χρήση μνήμης φαίνεται ότι βοηθάει στην εκπαίδευση του πράκτορα και επιταχύνει την εκπαίδευση ενώ η χρήση του δεύτερου δικτύου είναι απαραίτητη για τη σταθερή αύξηση των ανταμοιβών.

Μια βελτίωση που έχει προταθεί ακόμη στη βιβλιογραφία είναι η χρήση ενός ‘Dueling’ δικτύου. Το διαφορετικό βρίσκεται στην αρχιτεκτονική και συγκεκριμένα στο τελευταίο επίπεδο του δικτύου που αντί για να βγάζει έξοδο τη συνάρτηση μέγιστης χρησιμότητας q_* τη διαχωρίζει σε δύο κομμάτια: Τη συνάρτηση μέγιστης χρησιμότητας V(s) και το πλεόνασμα A(s,a). Το V δείχνει πόσο αναμενόμενη ανταμοιβή έχει η συγκεκριμένη κατάσταση, ενώ το πλεόνασμα δείχνει πόσο καλύτερη είναι η πραγματοποίηση της ενέργειας ‘a’ σε σχέση με όλες τις άλλες ενέργειες από τη κατάσταση s.

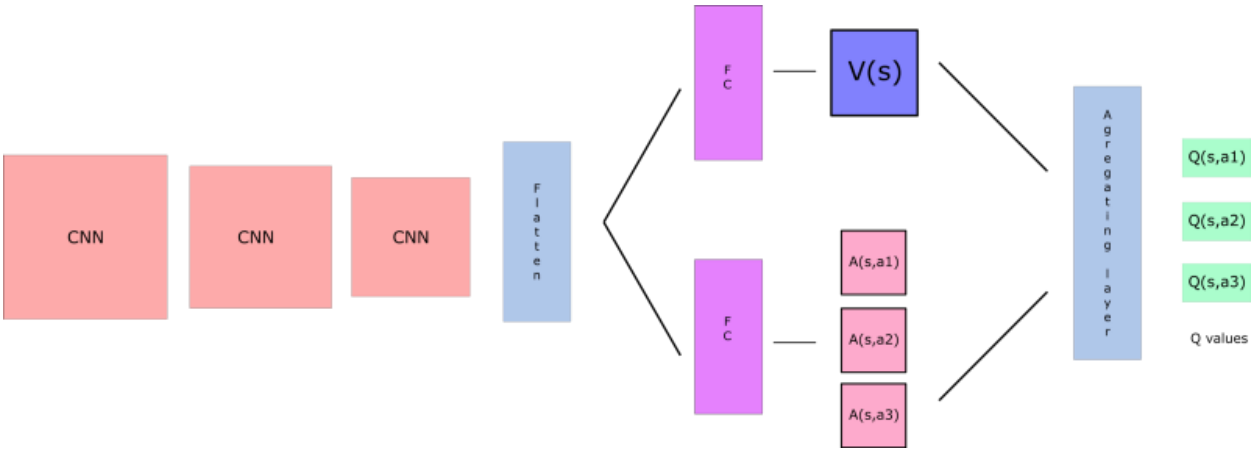


FIGURE 21 ΑΡΧΙΤΕΚΤΟΝΙΚΗ DUELING DEEP Q NETWORK

Η παραπάνω αρχιτεκτονική μέσου αυτού του διαχωρισμού της συνάρτησης χρησιμότητας έχει φανεί ότι επιταχύνει τη μάθηση και αυξάνει τις ανταμοιβές σε κάποιες περιπτώσεις χωρίς όμως να έχει αποδειχθεί.

Ο αλγόριθμος που παρουσιάστηκε μαζί με τις βελτιστοποιήσεις των δύο δικτύων και της μνήμης με προτεραιότητα ονομάζεται στη βιβλιογραφία και ως ‘Double Deep Q Learning with Priority Replay’ και είναι από τους πιο συχνά χρησιμοποιούμενους στα προβλήματα της ενισχυτικής μάθησης καθώς έχει βρει μεγάλη επιτυχία.

Ο επόμενος αλγόριθμος που παρουσιάζεται συνδέει δύο βασικές μεθόδους της ενισχυτικής μάθησης. Η πρώτη από αυτές που παρουσιάστηκε είναι αυτή της προσέγγισης της συνάρτησης μέγιστης χρησιμότητας ενέργειας. Η δεύτερη μέθοδος που είναι γνωστή ως Policy Gradient συνδέεται με την ίδια τη πολιτική την οποία προσπαθεί να παραμετροποιήσει και να βελτιώσει. Στην επόμενη ενότητα παρουσιάζεται μια σύντομη παρουσίαση για τις Policy Gradient μεθόδους και έπειτα μια παρουσίαση του A3C (Asynchronous Advantage Actor-Critic) αλγορίθμου ο οποίος συνδυάζει και τις δύο μεθόδους.

5 A3C Αλγόριθμος (Asynchronous Advantage Actor-Critic)

5.1 Εισαγωγικά

Η προηγούμενη ενότητα παρουσίασε έναν ισχυρό αλγόριθμος ο οποίος προσπαθεί να ανακαλύψει τη βέλτιστη πολιτική προσεγγίζοντας τη συνάρτηση μέγιστης χρησιμότητας ενέργειας.

Υπάρχει ακόμη μια μέθοδος ανακάλυψης της βέλτιστης συμπεριφοράς ενός πράκτορα η οποία βασίζεται στην παραμετροποίηση της ίδιας της πολιτικής. Αυτή η μέθοδος θα παρουσιαστεί στο πρώτο μέρος αυτής της ενότητας. Στη συνέχεια θα γίνει παρουσίαση του δεύτερου αλγορίθμου (Actor – Critic) ο οποίος συνδυάζει και τις δύο μεθόδους μάθησης.

Όπως και στο προηγούμενο αλγόριθμο, αρχικά θα γίνει μια παρουσίαση του ίδιο του αλγορίθμου, ύστερα θα παρουσιαστούν τα αποτελέσματα που έδωσε ο αλγόριθμος στα περιβάλλοντα του Atari και τέλος θα εκφράσω τις παρατηρήσεις και τα συμπεράσματα μου καθώς και μια σύγκριση των δύο αλγορίθμων.

5.2 Μέθοδοι Κλίσεων Πολιτικών (Policy Gradient Methods)

Μέχρι τώρα η πολιτική που ακολουθούσε ο πράκτορας ‘π’ χτιζόταν με βάση τη συνάρτηση μέγιστης ωφέλειας. Για παράδειγμα σε μια ε-άπληστη στρατηγική η πολιτική του πράκτορα ήταν η επιλογή της ενέργειας με τη μεγαλύτερη χρησιμότητα με πιθανότητα μεγαλύτερη του ε και μιας τυχαίας ενέργειας με πιθανότητα μικρότερη του ε.

Στις Policy Gradient μεθόδους η σκοπιά στρέφεται στην ίδια τη πολιτική. Αναλυτικότερα, γίνεται παραμετροποίηση της ίδιας της πολιτικής π_θ και εξαρτάται από τις παραμέτρους θ. Όπως και στο προηγούμενο αλγόριθμο θα γίνει η χρήση ενός νευρωνικού δικτύου το οποίο σε αυτή τη περίπτωση δέχεται ως είσοδο όπως και πριν τα χαρακτηριστικά της εισόδου αλλά αυτή τη φορά παράγει μια έξοδο για κάθε ενέργεια η οποία εκφράζει τη πιθανότητα να πραγματοποιήσει τη συγκεκριμένη ενέργεια από τη κατάσταση που βρίσκεται.

Επομένως, ο σκοπός του πράκτορα είναι να εκπαιδευτεί ώστε να παράγει μεγάλες πιθανότητες στις ενέργειες που του προσφέρουν τη μεγαλύτερη ανταμοιβή δηλαδή σε αυτές τις ενέργειες που θεωρούνται ‘καλές’ και μικρή πιθανότητα στις ενέργειες που του προσφέρουν λίγη ή καθόλου ανταμοιβή

('κακές' ενέργειες). Έπειτα, επιλέγει μια τυχαία ενέργεια από τις διαθέσιμες με την αντίστοιχη πιθανότητα με την έξοδο του δικτύου (τεχνική της ρουλέτας). Τα παραπάνω παρουσιάζονται στην εικόνα που ακολουθεί:

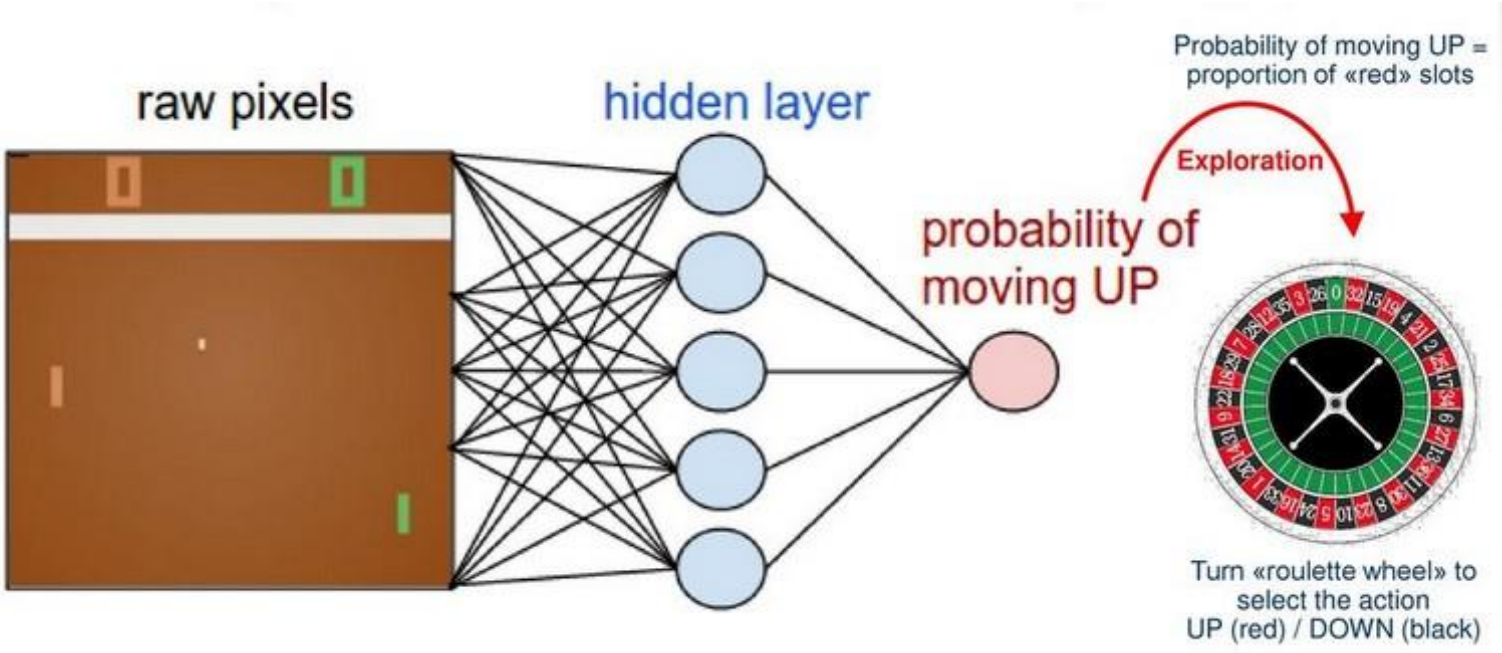


FIGURE 22 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΔΙΚΤΥΟΥ ΠΟΛΙΤΙΚΗΣ

Τώρα προκύπτει το ερώτημα για το πώς εκπαιδεύεται το δίκτυο ώστε να παράγει μικρές πιθανότητες στις 'κακές' κάθε φορά ενέργειες και μεγάλη πιθανότητα στις 'καλές'; Η λύση και σε αυτή τη περίπτωση είναι μέσω των ανταμοιβών που λαμβάνει ο πράκτορας και των επαναλαμβανόμενων επαναλήψεων.

Πιο συγκεκριμένα, μπορούμε να διαχωρίσουμε μια καλή στρατηγική σε σχέση με μια χειρότερη με βάση τη μέση αναμενόμενη ανταμοιβή που λαμβάνει ο πράκτορας στη διάρκεια ενός επεισοδίου. Η αναμενόμενη ανταμοιβή εξαρτάται άμεσα από τη πολιτική του πράκτορα και άρα από τις παραμέτρους 'θ' ή τα βάρη του νευρωνικού δικτύου. Επομένως, στόχος του πράκτορα είναι να μεγιστοποιήσει την συνάρτηση αναμενόμενης ανταμοιβής:

$$J(\theta) = E_{\pi}\left[G_t\right]$$

Αν ο πράκτορας καταφέρει να μεγιστοποιήσει την αναμενόμενη συνάρτηση ανταμοιβής δηλαδή να βρει τις παραμέτρους της πολιτικής 'θ' που τη μεγιστοποιούν θα έχει ανακαλύψει τη βέλτιστη πολιτική. Και σε αυτή τη περίπτωση μπορεί να γίνει η χρήση του διανύσματος κλίσεων $\nabla_{\theta} J(\theta)$ το οποίο

όπως αναφέρθηκε δείχνει τη κατεύθυνση της πιο απότομης αύξησης του J σε σχέση με τις παραμέτρους θ. Επειδή σε αυτή τη περίπτωση θέλουμε να μεγιστοποιήσουμε τη συνάρτηση αρκεί να αλλάζουμε τα βάρη θ με μικρά βήματα προς τη διεύθυνση του $\nabla_\theta J(\theta)$. Επομένως, η ανανέωση των βαρών του δικτύου προκύπτει ως:

$$\Delta\theta = \alpha * \nabla_\theta J(\theta)$$

Όπου ως 'a' ορίζεται και πάλι ένας ρυθμός μάθησης.

Θα περάσουμε τώρα την ανάλυση αυτού του διανύσματος κλίσεων. Αρχικά, αν με π_θ συμβολίσουμε τη συνάρτηση της πολιτικής δηλαδή τη συνάρτηση που υλοποιεί το νευρωνικό δίκτυο τότε το διάνυσμα κλίσεων αυτής ως προς τις παραμέτρους θ με τη χρήση μαθηματικών προκύπτει ως:

$$\nabla_\theta \pi_\theta(s,a) = \pi_\theta(s,a) * \frac{\nabla_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)} = \pi_\theta(s,a) * \nabla_\theta \log(\pi_\theta(s,a))$$

Αφού η παράγωγος του λογαρίθμου μιας σύνθετης συνάρτησης f(x) ορίζεται ως 1/f(x) επί τη παράγωγο του f(x). Η τελευταία εξίσωση ονομάζεται στη στατιστική και ως λόγος πιθανοφάνειας (likelihood ratio).

Αν το επεισόδιο διαρκούσε 1 βήμα τότε η συνολική επιστροφή θα ισούταν με R_t που είναι η άμεση ανταμοιβή μετά την πρώτη ενέργεια. Αν με P(s) συμβολίσουμε τη πιθανότητα ο πράκτορας να αρχίσει από κάποια κατάσταση s, τότε για αυτό την απλοϊκή διαδικασία απόφασης Markov του ενός βήματος η αναμενόμενη συνάρτηση ανταμοιβής αναλύεται ως:

$$J(\theta) = E_{\pi_\theta}[R_t] = \sum_{s \in S} P(s) * \sum_{a \in A} \pi_\theta(s,a) * R_{s,a}$$

Και η αντίστοιχη κλίση με βάση τα παραπάνω ισούται με:

$$\nabla J_\theta(\theta) = \sum_{s \in S} P(s) * \sum_{a \in A} \pi_\theta(s,a) * \nabla_\theta \log \pi_\theta(s,a) * R_{s,a}$$
$$= E_{\pi_\theta}[\nabla_\theta log \pi_\theta(s,a) * R_t]$$

Το θεώρημα κλίσεων πολιτικής (Policy Gradient Theorem) που έχει αποδειχθεί με τη χρήση μαθηματικών, επεκτείνει το παραπάνω θεώρημα για διαδικασίες απόφασης Markov πολλών βημάτων με την αλλαγή του άμεσης ανταμοιβής R_t, με τη συνολική συνάρτηση χρησιμότητας ενέργειας $q_\pi(s,a)$.

Δηλαδή:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) * q_{\pi_\theta}(s,a)]$$

Επομένως, από το παραπάνω θεώρημα προκύπτει και η ενημέρωση των βαρών σε κάθε βήμα του αλγορίθμου ως:

$$\Delta w = a * \nabla_\theta J(\theta) = \nabla_\theta \log \pi_\theta(s,a)\, q_{\pi_\theta}(s,a)$$

Στο παραπάνω θεώρημα βασίζονται όλες οι μέθοδοι που παραμετροποιούν τη κλίση. Η διαφοροποίηση των διαφόρων μεθόδων κλίσεων πολιτικής έγκειται στη διαδικασία υπολογισμού του q. Για παράδειγμα μια αμερόληπτη επιλογή του q θα μπορούσε να είναι η πραγματική επιστροφή ανταμοιβής στη διάρκεια ενός επεισοδίου. Δηλαδή συσσώρευση των ανταμοιβών και υπολογισμός τους q στο τέλος του επεισοδίου ως το άθροισμα των προεξοφλημένων αυτών ανταμοιβών. Αυτός ο αλγόριθμος είναι γνωστός και ως REINFORCE και είναι από τους πρώτους που έχουν προταθεί από αυτή τη γκάμα αλγορίθμων. Αν και σε απλά προβλήματα μπορεί να αποδώσει δεν είναι αρκετά καλός για τα πιο σύνθετα προβλήματα όπως του Atari. Αυτό το πρόβλημα λύνει ο αλγόριθμος Actor – Critic που παρουσιάζεται στην επόμενη ενότητα και χρησιμοποιεί όσα ειπώθηκαν στη προηγούμενη ενότητα για τον υπολογισμό της αναμενόμενης ανταμοιβής Q.

5.3 Παρουσίαση Actor – Critic Αλγορίθμου

5.3.1 Απλός Actor – Critic

Ο Actor-Critic (Ενεργής- Κριτής) αλγόριθμος είναι ένας υβριδικός αλγόριθμος μεταξύ των αλγορίθμων προσέγγισης της συνάρτησης χρησιμότητας που παρουσιάστηκαν στην ενότητα 4 και των αλγορίθμων κλίσεων πολιτικών. Αυτή του η φύση έγκειται στη χρήση δύο διαφορετικών νευρωνικών δικτύων, τον Actor και τον Critic και τη προσπάθεια συνεργασίας τους για την ολοκλήρωση του στόχου.

Αναλυτικότερα, ο κριτής (Critic) προσπαθεί να προβλέψει τη συνάρτηση μέγιστης ωφέλειας q(s,a) ή v(s). Πρόκειται λοιπόν για ένα νευρωνικό δίκτυο με βάρη w όπως αυτού που παρουσιάστηκε στην ενότητα 4 για τη προσέγγιση της συνάρτησης μέγιστης ωφέλειας κατάστασης ή ενέργειας $v_\pi(s)$ ή $q_\pi(s,a)$. Η προσέγγιση της συνάρτησης χρησιμότητας δεν κάνει χρήση κάποιας μνήμης εμπειριών στη προκειμένη περίπτωση άλλα εκπαιδεύεται απευθείας στις εμπειρίες που λαμβάνει ο πράκτορας.

Έπειτα, το νευρωνικό δίκτυο του Actor πρόκειται για ένα νευρωνικό δίκτυο όπως αυτών που περιγράφηκαν στην προηγούμενη υπο-ενότητα, δηλαδή ενός δικτύου που παίρνει ως είσοδο τη κατάσταση του περιβάλλοντος και προβλέπει για κάθε ενέργεια τη πιθανότητα να την ακολουθήσει. Το νευρωνικό δίκτυο του Actor λαμβάνει υπόψη την αναμενόμενη χρησιμότητα που βγάζει ως έξοδο ο Critic για τη συγκεκριμένη κατάσταση για την ενημέρωση των βαρών του θ. Έτσι, από το θεώρημα κλίσεων πολιτικής αν συμβολίσουμε με $q_w(s,a)$ την έξοδο του κριτή για το συγκεκριμένο ζεύγος κατάστασης-ενέργειας η ενημέρωση των βαρών του Actor είναι:

$$\Delta\theta = a * \nabla_\theta \log \pi_\theta(s,a) * q_w(s,a)$$

Ο πράκτορας λαμβάνει την επόμενη ενέργεια του με βάση τις πιθανότητες για κάθε πιθανή ενέργεια στην έξοδο του Actor (τεχνική της ρουλέτας). Και τα δύο δίκτυα εκπαιδεύονται ταυτόχρονα και προσπαθούν να ανακαλύψουν την καλύτερη στρατηγική π_θ μέσω της προσέγγισης της συνάρτησης μέγιστης ωφέλειας q_w. Με αυτό το τρόπο ο αλγόριθμος Actor-Critic προσπαθεί να συνδυάσει τα θετικά και από τις δύο κλάσεις αλγορίθμων για την εκπαίδευση του πράκτορα. Ο παραπάνω τρόπος αποτελεί το σκελετό του Actor-Critic

αλγορίθμου. Για τη παραπέρα βελτίωσή του έχουν προταθεί κάποιες περεταίρω προσθήκες στη δομή του αλγορίθμου.

5.3.2 Πλεονασματικός Actor-Critic (A2C)

Ο παραπάνω αλγόριθμος, λόγω της χρήσης ενός δικτύου για τη πρόβλεψη της συνάρτησης ωφέλειας q_w καταλήγει να έχει πολύ μεγάλη αστάθεια η οποία ενδέχεται να παρουσιάσει προβλήματα. Η πρώτη προσθήκη έχει ως στόχο τη καταπολέμηση αυτής της αστάθειας από το δίκτυο. Το πετυχαίνει αυτό με την αφαίρεση μιας συνάρτησης αναφοράς B από τη συνάρτηση ωφέλειας ενέργειας $q_w(s,a)$. Μια καλή επιλογή του B είναι η συνάρτηση χρησιμότητας κατάστασης v(s) η οποία μειώνει δραματικά την αστάθεια του δικτύου. Αυτό όμως θα προϋπέθετε τη χρήση ενός νέου νευρωνικού δικτύου που θα προβλέπει τη συνάρτηση χρησιμότητας $v_v(s)$ για μια κατάσταση s με βάρη v. Η χρήση τριών διαφορετικών νευρωνικών δικτύων όμως επιβραδύνει αρκετά τους χρόνους εκτέλεσης καθώς απαιτεί και περισσότερο συγχρονισμό.

Η λύση σε αυτό το πρόβλημα προέρχεται από το γεγονός ότι το σφάλμα της πραγματικής συνάρτησης ωφέλειας V(s) έχει αποδειχθεί ότι είναι μια αμερόληπτη εκτίμηση της συνάρτησης πλεονάσματος A = q(s,a) – v(s) δηλαδή αν με ‘δ’ συμβολίσουμε το σφάλματα της συνάρτησης ωφέλειας:

$$\delta = R_t + \gamma * v(s') - v(s)$$

Και από τα παραπάνω:

$$E[\delta|s,a] = E[R_t + \gamma * v(s')|s,a] - V(s) = q(s,a) - v(s) = A(s,a)$$

Έτσι έχει εξαλειφτεί η ανάγκη δύο δικτύων για τον υπολογισμό της συνάρτησης πλεονάσματος και αρκεί μόνο ένα δίκτυο (Κριτής) που θα προβλέπει τη συνάρτηση ωφέλειας $v_w(s)$ και μέσω του σφάλματος αυτής θα υπολογίζεται η συνάρτηση πλεονάσματος. Έπειτα, αυτή η συνάρτηση πλεονάσματος αντικαταστεί τη συνάρτηση ωφέλειας ενέργειας q στην ενημέρωση των βαρών του Actor. Με αυτό το τρόπο ο Actor ενημερώνει τα βάρη του με το παραπάνω κανόνα:

$$\Delta\theta = a * \nabla_\theta \log \pi_\theta(s,a) * \boldsymbol{A}(\boldsymbol{s},\boldsymbol{a})$$

Ο αλγόριθμος αυτός με τη παραπάνω προσθήκη είναι γνωστός και ως Πλεονασματικός Actor-Critic (Advantage Actor Critic-A2C) και από πολλούς θεωρείται ένας από τους καλύτερους αλγορίθμους της ενισχυτικής μάθησης.

5.3.3 Ασύγχρονος Πλεονασματικός Actor-Critic (A3C)

Η τελευταία προσθήκη στο παραπάνω αλγόριθμο αν και προαιρετική επιταχύνει πολύ την εκπαίδευση και παράγει καλύτερα αποτελέσματα. Αυτό που κάνει είναι διατηρώντας την φιλοσοφία του πλεονασματικού Actor-Critic να εισάγει παραλληλισμό στην εκτέλεση.

Πιο συγκεκριμένα, παρατηρήθηκε ότι η εμπειρία που συλλέγει ο Actor από το περιβάλλον μπορεί να παραλληλοποιηθεί με τη προσθήκη περισσότερων Actors. Αυτοί οι Actors έχουνε ο καθένας ένα ξεχωριστό τοπικό δίκτυο πολιτικής και δίκτυο προσέγγισης συνάρτησης ωφέλειας καθώς και αλληλοεπιδρούν ο καθένας με ένα ξεχωριστό αντίγραφο του περιβάλλοντος. Ο συντονισμός όλων των Actors επιτυγχάνεται με τη χρήση ενός μοναδικού καθολικού δικτύου το οποίο ενημερώνει τα βάρη του ανά τακτά χρονικά διαστήματα με βάση το διάνυσμα κλίσεων (κανόνα δέλτα) των τοπικών δικτύων. Έπειτα, κάθε actor ενημερώνει τα βάρη του με βάση αυτά του συντονιστή καθολικού δικτύου.

Τα παραπάνω παρουσιάζονται στην εικόνα που ακολουθεί:

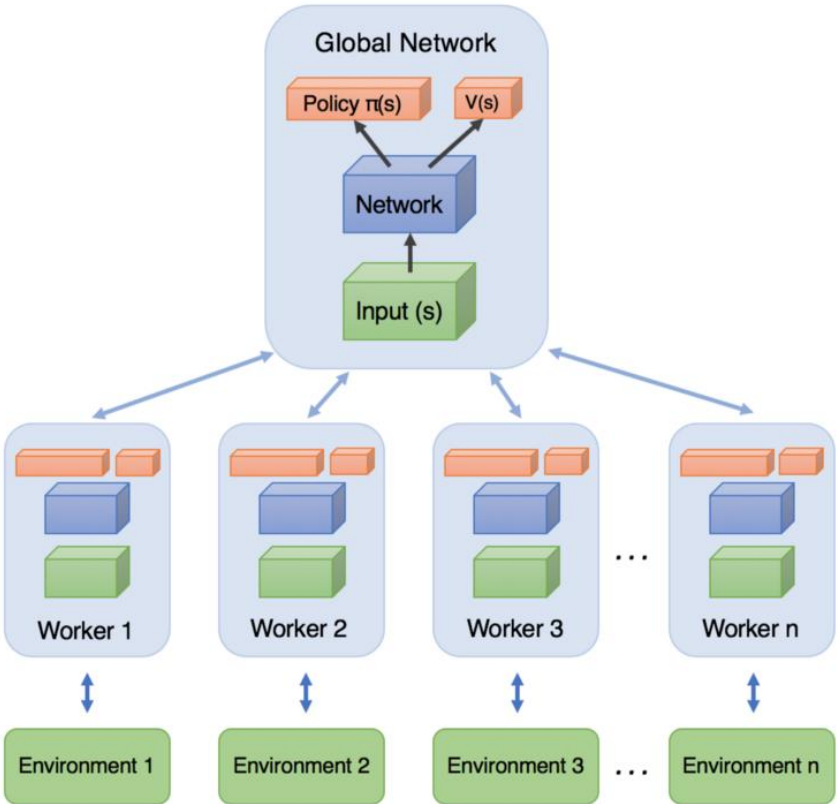


FIGURE 23 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΑΣΥΧΡΟΝΟΥ ACTOR CRITIC

Κάθε Worker αντιστοιχεί σε έναν Actor (νήματα) που αλληλοεπιδρά με βάση των τοπικών του δικτύων με το τοπικό του περιβάλλον και ενημερώνει τα βάρη του καθολικού δικτύου με βάση τις εμπειρίες του. Τέλος, ανά τακτά χρονικά διαστήματα συντονίζει τα τοπικά του βάρη με αυτά του καθολικού δικτύου.

Ο λόγος που ο ασύγχρονος αλγόριθμος Πλεονασματικός Actor-Critic είναι τόσο ισχυρός και έχει βρει τεράστια επιτυχία (εκτός της αυξημένης ταχύτητας που προσφέρει) έγκειται στις διαφορετικές εμπειρίες που λαμβάνει κάθε Actor από το περιβάλλον του. Ο αρχικός Actor-Critic παρουσίαζε αρκετά προβλήματα λόγω το ότι κάθε εμπειρία στην οποία εκπαιδευόταν ο πράκτορας ήταν πολύ στενά συνδεδεμένη με τις υπόλοιπες με αποτέλεσμα να είναι ιδιαίτερα ασταθής ή να σταματάει πρόωρα την εκπαίδευση λόγω υπερπροσαρμογής στις εμπειρίες-δεδομένα. Με τον ασύγχρονο Actor-Critic κάθε πράκτορας αλληλοεπιδρά διαφορετικά με το δική του αντίγραφο του περιβάλλοντος με αποτέλεσμα να παράγουν διαφορετικές εμπειρίες. Η εκπαίδευση του καθολικού δικτύου σε αυτή την γκάμα ανεξάρτητων εμπειριών εξαλείφει τελείως το πρόβλημα και ο πράκτορας εκπαιδεύεται πολύ πιο αποδοτικά.

5.4 Υλοποίηση Ασύγχρονου Πλεονασματικού Actor-Critic (A3C)

Η υλοποίηση στη συγκεκριμένη πτυχιακή εργασία χρησιμοποιεί ένα κοινό νευρωνικό δίκτυο για την πολιτική και για τη προσέγγιση μέγιστης ωφέλειας αντί για δύο ξεχωριστά καθώς και μια κοινή συνάρτηση σφάλματος J(θ) που το δίκτυο προσπαθεί να ελαχιστοποιήσει.

Αναφορικά με το κοινό δίκτυο, ο λόγος που είναι δυνατό να γίνει αυτό είναι το ότι τα κατώτερα επίπεδα του δικτύου εκπαιδεύονται να αναγνωρίσουν διάφορα πρότυπα στην εικόνα και επομένως συνίστανται να είναι κοινά. Το τελευταίο κρυφό επίπεδο λαμβάνοντας υπόψη την έξοδο των προηγούμενων κρυφών επιπέδων μεταφέρει την έξοδο του με διαφορετικά βάρη σε δύο ξεχωριστούς τύπους εξόδων. Ο πρώτος αποτελείται από έναν νευρώνα και είναι αυτός που βγάζει ως έξοδο τη συνάρτηση χρησιμότητας κατάστασης με γραμμική συνάρτηση ενεργοποίησης. Η δεύτερη κατηγορία αποτελείται από τόσες εξόδους όσες και οι δυνατές ενέργειες στο συγκεκριμένο περιβάλλον και δείχνουν τις πιθανότητες κάθε ενέργειας, πρόκειται δηλαδή για τη πολιτική του πράκτορα.

Για να δουλέψει το παραπάνω πρέπει να γίνει επίσης η χρήση μιας κοινής συνάρτησης σφάλματος που λαμβάνει υπόψη και το σφάλμα της πολιτικής καθώς και το σφάλμα της συνάρτησης προσέγγισης.

Το σφάλμα της πολιτικής για ένα συγκεκριμένο ζεύγος κατάστασης-ενέργειας είναι ίδιο με το σφάλμα της συνάρτησης πολιτικών που παρουσιάστηκε στη προηγούμενη ενότητα δηλαδή:

$$P_{\theta\,loss} = \log(\pi_\theta(a|s) * A(s,a)$$

Το σφάλμα για τη προσέγγιση της συνάρτησης ωφέλειας κατάστασης v(s) αν υποθέσουμε ότι υπολογίζεται κάθε ένα βήμα μετά από ανταμοιβή R_t είναι ίσο με το μέσο τετραγωνικό σφάλμα (πρόβλημα παλινδρόμησης):

$$V_{\theta\,loss} = \left(\pi\rho\alpha\gamma\mu\alpha\tau\iota\kappa\acute{\eta}_{\tau\iota\mu\acute{\eta}} - \pi\rho\acute{o}\beta\lambda\epsilon\psi\eta\right)^2 = \left(R_t + \gamma * v_\theta(s') - v_\theta(s)\right)^2$$
$$= A(s,a)$$

Τέλος, έχει προταθεί η αφαίρεση του σφάλματος της εντροπίας από τη συνάρτηση σφάλματος για την αποφυγή πρόωρων τοπικών ελαχίστων. Η εντροπία ορίζεται ως:

$$E_{loss}(\theta) = -\sum_{a \in A} \pi_\theta(a|s) * \log(\pi_\theta(a|s))$$

Και στο πλαίσιο της μηχανικής μάθησης χρησιμοποιείται ως μετρική για τη τυχαιότητα ή σε μια κατανομή πιθανοτήτων. Η εντροπία παίρνει μεγάλες τιμές όταν η κατανομή πιθανοτήτων αποτελείται από πολύ κοντινές πιθανότητες ενώ μικρή όταν αποτελείται από πιθανότητες με μεγάλη διαφορά. Επειδή αφαιρείται από τη συνολική συνάρτηση σφάλματος την οποία προσπαθούμε να ελαχιστοποιήσουμε στόχος είναι η μεγιστοποίησή της. Αυτό είναι ιδιαίτερα σημαντικό στα πρώτα βήματα του αλγορίθμου καθώς ενθαρρύνει τις ισοπίθανες κατανομές στις ενέργειες του πράκτορα με αποτέλεσμα να παροτρύνει το πράκτορα να εξερευνήσει το περιβάλλον του. Αυτό έχει ως αποτέλεσμα την αποφυγή πολύ πρόωρων τοπικών ελαχίστων που μπορεί εύκολα να κολλήσει ο πράκτορας κυρίως στα αρχικά βήματα της εκπαίδευσης και στα οποία πραγματοποιεί μονάχα μια ενέργεια με πιθανότητα 1.

Σύμφωνα με όλα τα παραπάνω η συνολική συνάρτηση σφάλματος που ο πράκτορας προσπαθεί να ελαχιστοποιήσει προκύπτει ως:

$$J_{total}(\theta) = \beta_1 * V_{loss} - \beta_2 * E_{loss} - \beta_3 * P_{loss}$$

Ο λόγος που αφαιρείται το σφάλμα της πολιτικής από τη συνολική συνάρτηση σφάλματος είναι ο ίδιο λόγος που αναφέρθηκε και για την εντροπία δηλαδή το ότι η βέλτιστη πολιτική πετυχαίνετε όταν η συνάρτηση σφάλματος της πολιτικής μεγιστοποιείται και για αυτό γίνεται η αφαίρεσή της από τη συνολική συνάρτηση σφάλματος αφού πρόκειται για ένα πρόβλημα ελαχιστοποίησης.

Τα β_1,$\beta_2\beta_3$ είναι τα αντίστοιχα βάρη που πολλαπλασιάζονται με τις επιμέρους συναρτήσεις σφάλματος, ρυθμίζουν το πόσο βαρύτητα και επομένως πόσο μερίδιο κατέχει ένα συγκεκριμένο σφάλμα στον υπολογισμό του συνολικού σφάλματος.

Το δίκτυο που χρησιμοποίησα στο συγκεκριμένο αλγόριθμο πρόκειται πάλι για ένα συνελεκτικό δίκτυο αλλά έχει κάποιες διαφοροποιήσεις στην αρχιτεκτονική του καθώς το δίκτυο της προηγούμενης ενότητας δεν εκπαιδευόταν επαρκώς στο συγκεκριμένο αλγόριθμο.

Παρακάτω παρουσιάζεται αναλυτικά η αρχιτεκτονική του δικτύου:

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΔΙΚΤΥΟΥ:

1ο Συνελεκτικό Επίπεδο: 32 πυρήνες μεγέθους 5x5, γέμισμα='same', ενεργοποίηση = 'relu'

Επίπεδο MaxPool: μέγεθος 2x2

2ο Συνελεκτικό Επίπεδο: 32 πυρήνες μεγέθους 5x5, γέμισμα='same', ενεργοποίηση = 'relu'

Επίπεδο MaxPool: μέγεθος 2x2

3ο Συνελεκτικό Επίπεδο: 64 πυρήνες μεγέθους 4x4, γέμισμα='same', ενεργοποίηση = 'relu'

Επίπεδο MaxPool: μέγεθος 2x2

4ο Συνελεκτικό Επίπεδο: 64 πυρήνες μεγέθους 3x3, γέμισμα='same', ενεργοποίηση = 'relu'

Πλήρως συνδεδεμένο επίπεδο: 512 νευρώνες, συνάρτηση ενεργοποίησης: 'relu'

Έξοδος Συνάρτησης Ωφέλειας: 1 νευρώνας, συνάρτηση ενεργοποίησης=γραμμική

Έξοδος Πολιτικής: Πλήθος Νευρώνων όσες οι διαθέσιμες ενέργειες, συνάρτηση, χωρίς συνάρτηση ενεργοποίησης

Το padding/ γέμισμα 'same' σημαίνει γεμίζει την εικόνα στα όρια για μηδενικά ώστε να μετά την εφαρμογή κάθε πυρήνα να διατηρήσει το αρχικό της μέγεθος. Το MaxPool επίπεδο που προστίθεται μετά από κάθε συνελεκτικό επίπεδο ουσιαστικά μειώνει το πλήθος των εξόδων των πυρήνων διαλέγοντας σε κάθε 2x2 τετράγωνο το μέγιστο στοιχείο. Αυτό μειώνει κατά πολύ το πλήθος των βαρών ενώ ρυθμίζει ταυτόχρονα το overfitting.

Το παραπάνω δίκτυο εκπαιδεύτηκε με τον αλγόριθμο-ψευδοκώδικα που παρουσιάζεται παρακάτω με τη μέθοδο Adam (παραλλαγή Gradient Descent) με στόχο την ελαχιστοποίηση της συνολικής συνάρτησης σφάλματος J(θ) που παρουσιάστηκε παραπάνω.

Ο αλγόριθμος μάθησης A3C είναι ως εξής:

Αλγόριθμος A3C:

Κατασκευή του καθολικού δικτύου

Δημιουργία N νημάτων, N τοπικών δικτύων και N περιβαλλόντων για αυτά τα δίκτυα

Αρχικοποίηση των τοπικών δικτύων με τα βάρη του καθολικού

Έναρξη Νημάτων

Αναμονή μέχρι να τελειώσουν

Αλγόριθμος για κάθε νήμα actor-critic:

Αρχικοποίηση πινάκων κατάστασης, ανταμοιβής, ενέργειας

Για 1 έως K επισόδεια:

Αρχικοποίηση του περιβάλλοντος και αποθήκευση αρχικής κατάστασης s

Για 1 μέχρι N βήματα και όσο δεν έχει τελειώσει το επεισόδιο:

Επιλογή κατάστασης με βάση τοπικό δίκτυο και αντίστοιχές πιθανότητες στην έξοδο (τεχνική της ρουλέτας)

Πραγματοποίηση ενέργειας a, παρακολούθηση άμεσης ανταμοιβή r, επόμενης κατάστασης s'

Αποθήκευση s,r,a στις αντίστοιχές λίστες

Υπολογισμός προ εξοφλημένων ανταμοιβών στα N βήματα από πίνακα ανταμοιβών, όπου η τελευταία προ εξοφλημένη ανταμοιβή DR_t από τη λίστα ανταμοιβών ισούται με $R_t + \gamma * v(s_{t+1})$, $DR_{t-1} = R_{t-1} + \gamma * DR_t$ και ούτω κάθε εξής. v(s') = 0 αν έχει τελειώσει το επεισόδιο, αλλιώς υπολογίζεται από το τοπικό δίκτυο.

Υπολογισμός πλεονάσματος για κάθε στοιχείο της λίστας $A_t = DR_t - v_\vartheta(s_t)$

Πρόβλεψη για κάθε κατάσταση s_t στη λίστα όλων των πιθανοτήτων $\pi_\vartheta(s,a)$ για το τοπικό δίκτυο.

Υπολογισμός εντροπίας με βάση τις παραπάνω πιθανότητες

Υπολογισμός σφάλματος πολιτικής με βάση τις εξόδους του δικτύου και το πλεόνασμα.

Υπολογισμός σφάλματος συνάρτησης ωφέλειας ως A^2

Υπολογισμός της κλίσης της συνολικής συνάρτησης σφάλματος για το τοπικό δίκτυο

Λήψη κλειδώματος Lock

Εφαρμογή κλίσεων (κανόνας δέλτα) για τα βάρη του καθολικού δικτύου

Ενημέρωση βαρών τοπικού δικτύου με αυτά του καθολικού

Απελευθέρωση κλειδώματος unlock

Εκκαθάριση λιστών κατάστασης, ενέργειας, ανταμοιβής

Έτσι αν ο παραπάνω αλγόριθμος εφαρμοστεί για ένα μεγάλο αριθμό βημάτων το καθολικό δίκτυο θα εκπαιδευτεί δηλαδή θα αρχίσει να προσεγγίζει τη πραγματική μέγιστη συνάρτηση ωφέλειας και τη πραγματική μέγιστη πολιτική με αποτέλεσμα να συμπεριφέρεται βέλτιστα.

Πριν περάσουμε στα αποτελέσματα του συγκεκριμένου αλγορίθμου να σημειωθεί ότι στη δικιά μου υλοποίηση το καθολικό δίκτυο και τα τοπικά actor-critic αντί να βγάζουν για έξοδο πιθανότητες μέσω της συνάρτησης ενεργοποίησης ‘softmax’ βγάζουν την έξοδο τους χωρίς συνάρτηση ενεργοποίησης και έπειτα πρέπει να εφαρμοστεί ‘softmax’ για την ανάκτηση των πιθανοτήτων. Αυτό έγινε χάριν ευκολίας υπολογισμού της συνάρτησης σφάλματος μέσω των έτοιμων μεθόδων που παρέχει το tensorflow για τον υπολογισμό της εντροπίας καθώς και το σφάλμα της πολιτικής.

5.5 Αποτελέσματα Εκπαίδευσης

5.5.1 Παράμετροι

Ο παραπάνω αλγόριθμος εκπαιδεύτηκε με τις παρακάτω παραμέτρους στα ίδια περιβάλλοντα με τον double deep q learning αλγόριθμο. Παρουσιάζονται παρακάτω τα αποτελέσματα που έδωσε μετά την εκτέλεση:

Κοινές Παράμετροι (Όλα τα περιβάλλοντα)

Ρυθμός Προεξόφλησης = 0.99

Αριθμός από νήματα (actors) = 8

Αριθμός βημάτων για ενημέρωση καθολικού = 20

Συντελεστή βαρύτητας σφάλματος v $\beta_1 = 0.5$

Συντελεστής βαρύτητας σφάλματος entropy $\beta_2 = 0.01$

Συντελεστής βαρύτητας σφάλματος πολιτικής $\beta_3 = 1$

Μέθοδος Εκπαίδευσης: Adam

5.5.2 CartPole-v0

Χρησιμοποιήθηκε και πάλι το περιβάλλον CartPole-v0 για τους σκοπούς του testing, για να διαπιστώσω αν λειτουργεί ο αλγόριθμος. Λόγω ότι η είσοδός του δεν είναι εικόνα (στιγμιότυπο της οθόνης) έγινε χρήση ενός κρυφού πλήρως συνδεδεμένου επιπέδου με 24 νευρώνες και συνάρτηση ενεργοποίησης τη Relu. Τέλος, ο ρυθμός μάθησης για το συγκεκριμένο περιβάλλον είναι σταθερός και ίσος με 0.00025.

Οι ανταμοιβές του αλγορίθμου για 31 χιλιάδες επεισόδια στη φάση της εκπαίδευσης παρουσιάζονται στο παρακάτω διάγραμμα:

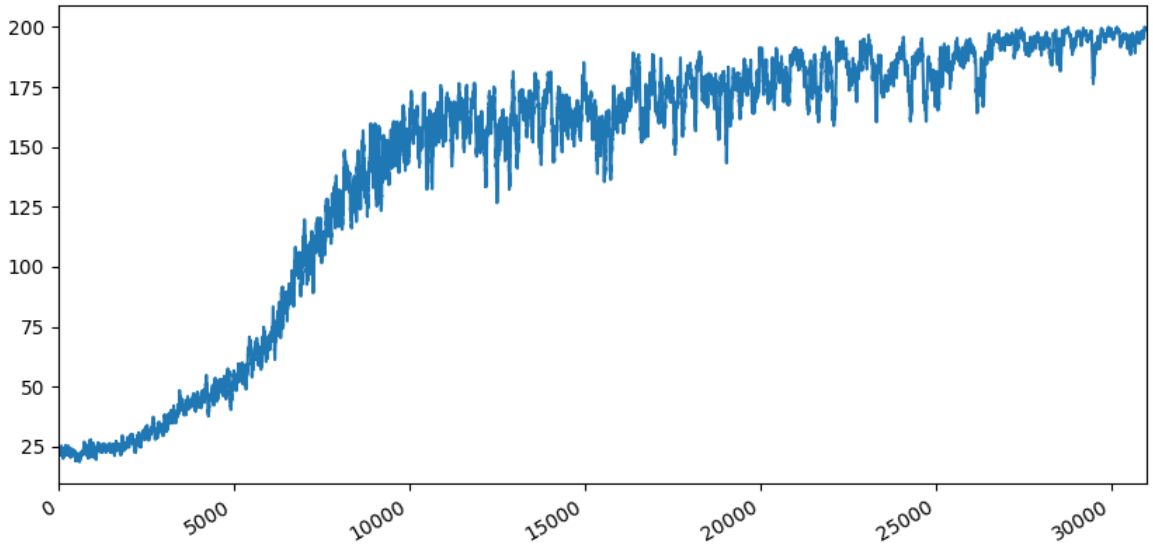


FIGURE 24 ΑΝΤΑΜΟΙΒΕΣ ΣΤΗ ΦΑΣΗ ΤΗΣ ΕΚΠΑΙΔΕΥΣΗΣ A3C - ΠΕΡΙΒΑΛΛΟΝ CARTPOLE

Βλέπουμε ότι ο πράκτορας κατάφερε να βρει τη βέλτιστη πολιτική να ισορροπεί δηλαδή το στύλο για 200 συνεχόμενα βήματα. Περνάμε τώρα στα πιο δύσκολα περιβάλλοντα του Atari.

Τα αποτελέσματα με τα βάρη που βρήκε στη φάση testing για 100 επεισόδια παρουσιάζονται παρακάτω:

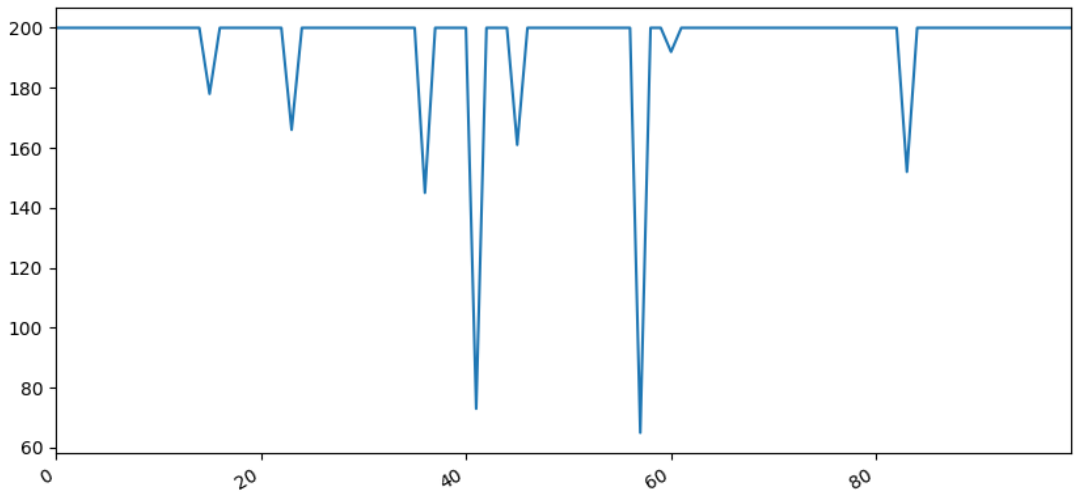


FIGURE 25 ΑΝΤΑΜΟΙΒΕΣ ΣΤΗ ΦΑΣΗ TESTING A3C - ΠΕΡΙΒΑΛΛΟΝ CARTPOLE

Όπως φαίνεται ο πράκτορας με κάποιες μικρές εξαιρέσεις καταφέρνει να πετύχει τη μέγιστη ανταμοιβή. Θα μπορούσε λόγω της απλοϊκής του φύσης να χρησιμοποιηθεί ένας μεγαλύτερος ρυθμός μάθησης.

5.5.3 Pong-v0

Στο Pong καθώς και στα υπόλοιπα Atari περιβάλλοντα που παρουσιάζονται έγινε χρήση του συνελεκτικού δικτύου που περιάφτηκε στη προηγούμενη ενότητα. Ο ρυθμός μάθησης σε αυτή τη περίπτωση είναι σταθερός σε όλη τη προπόνηση και ίσος με 0.0001. Επίσης, κάθε νήμα εξερευνάει το περιβάλλον του με πιθανότητα 0.7 και γραμμικά στα πρώτα 10 χιλιάδες βήματα αυτό μειώνεται στο μηδέν. Η αλήθεια είναι λόγω της αρχικοποίησης και της εντροπίας το δίκτυο από τη φύση του έχει τη τάση στα αρχικά βήματα να εξερευνάει και για αυτό δεν είναι απαραίτητο. Παρόλα αυτά αυτό ο ρυθμός εξερεύνησης παρατήρησα ότι επιτάχυνε τη βελτίωση της πολιτικής στα πρώτα βήματα του αλγορίθμου.

Στο παρακάτω διάγραμμα φαίνονται τα αποτελέσματα των ανταμοιβών του πράκτορα στο περιβάλλον Pong στη φάση της εκπαίδευσης για 3000 επεισόδια:

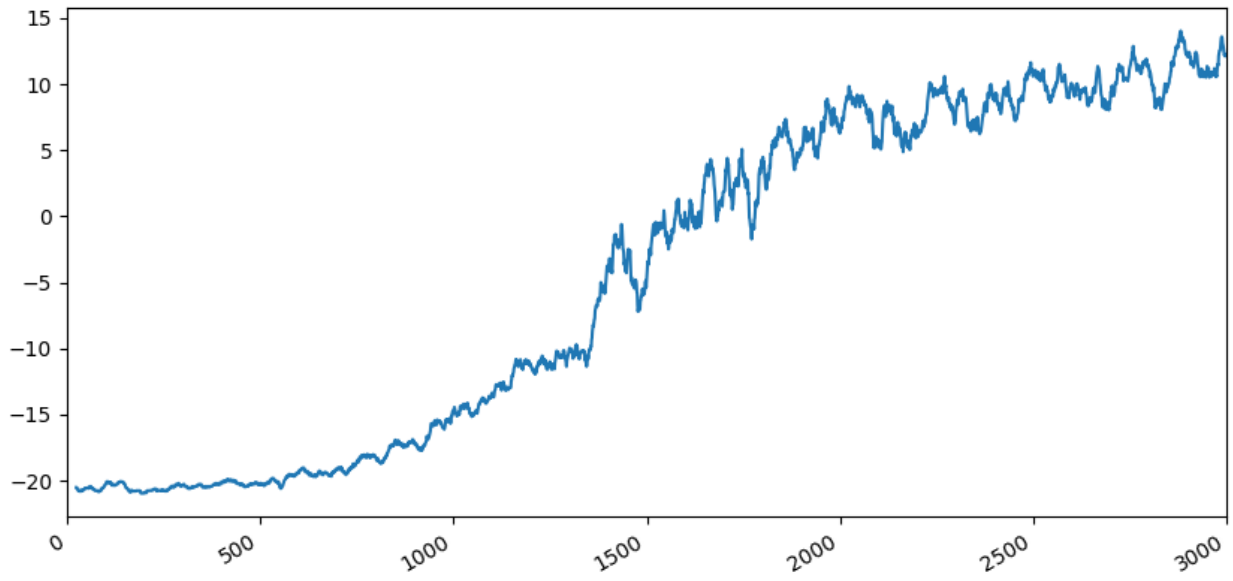


FIGURE 26 ΑΝΤΑΜΟΙΒΕΣ ΣΤΗ ΦΑΣΗ ΤΗΣ ΕΚΠΑΙΔΕΥΣΗΣ A3C - ΠΕΡΙΒΑΛΛΟΝ ATARI - PONG

Όπως μπορούμε να διακρίνουμε και σε αυτή τη περίπτωση οι ανταμοιβές του πράκτορα παρουσιάζουν μια καθαρά αυξητική πορεία. Ενώ ξεκίνησε με το να χάνει με διαφορά 21 με 20 πόντους στο τέλος της εκπαίδευσης καταφέρνει να νικάει με διαφορά κατά μέσο όρο 14-15. Ίσως αν συνεχιζόταν η εκπαίδευση για μεγαλύτερο αριθμό βημάτων ή αν και σε αυτή τη περίπτωση γινόταν χρήση μεταβλητού ρυθμού μάθησης να κατάφερνε μέση ανταμοιβή 20-21 όπως ο Double Deep Q Learning αλγόριθμος όμως κάτι τέτοιο δεν ήταν εφικτό λόγω έλλειψης χρόνου.

Στο παρακάτω διάγραμμα παρουσιάζονται οι ανταμοιβές του εκπαιδευμένου πράκτορα για 100 επεισόδια στη φάση του testing:

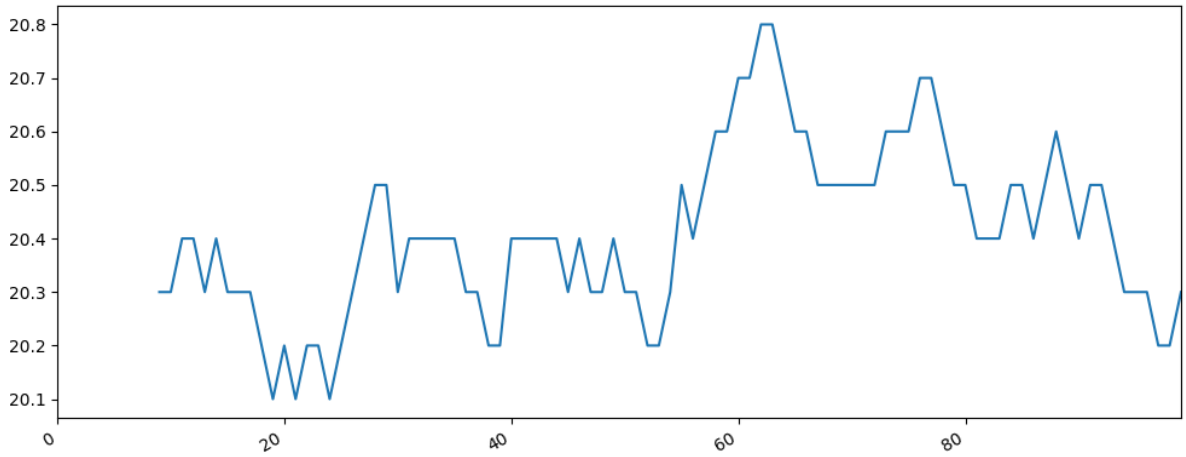


FIGURE 27 ΑΝΤΑΜΟΙΒΕΣ ΣΤΗ ΦΑΣΗ TESTING A3C - ΠΕΡΙΒΑΛΛΟΝ PONG

Όπως μπορούμε να δούμε ο πράκτορας έχει εκπαιδευτεί να λαμβάνει τιμές στο διάστημα [20, 21] και άρα έχει προσεγγίσει πολύ τη βέλτιστη πολιτική.

5.5.4 Super Mario Bros 1, Πίστα 1-1

Στο περιβάλλον του Super Mario έγινε προσπάθεια μιας και πρόκειται για πιο περίπλοκο περιβάλλον να γίνει η χρήση ενός μεταβλητού ρυθμού μάθησης. Κάθε νήμα έχει έναν αρχικό ρυθμό μάθησης ίσο με 0.0003 των οποίο μειώνουν γραμμικά στο 1e-20 σε 15 εκατομμύρια βήματα. Η συγκεκριμένη προσέγγιση είχε πού θετικά αποτελέσματα καθώς οι μεταβολές των βαρών στην αρχή της

προπόνησης ήταν πιο έντονες και απότομες, ενώ όσο προχωρούσε και προσέγγιζε το ελάχιστο μειώνονταν ώστε να μην αποκλίνει από αυτό.

Στο παρακάτω διάγραμμα παρουσιάζονται οι ανταμοιβές του πράκτορα στη διάρκεια εκπαίδευσης των 10 χιλιάδων επεισοδίων:

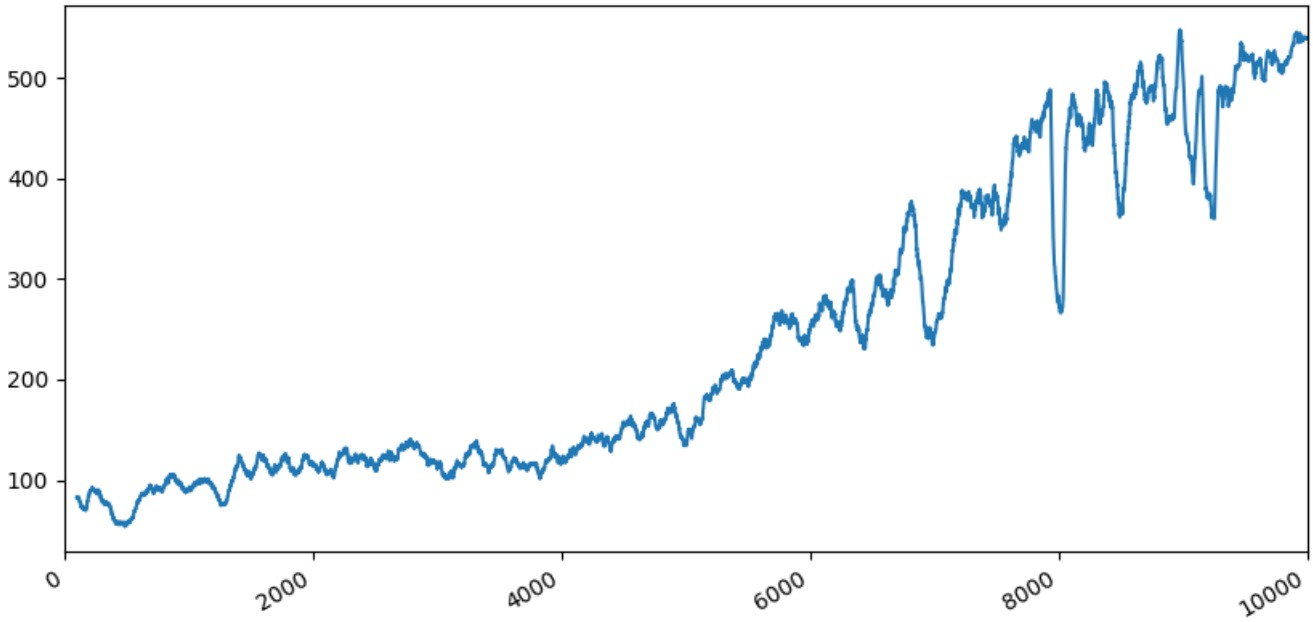


FIGURE 28 ΑΝΤΑΜΟΙΒΕΣ ΣΤΗ ΦΑΣΗ ΤΗΣ ΕΚΠΑΙΔΕΥΣΗΣ A3C - ΠΕΡΙΒΑΛΛΟΝ SUPER MARIO 1-1

Όπως φαίνεται από την παραπάνω εικόνα οι ανταμοιβές του παρουσιάζουν αυξητική πορεία και στη περίοδο των 10000 βημάτων κατάφερε τελικά να βρει μια βέλτιστη πολιτική στην οποία τερματίζει την πίστα και μάλιστα σε καλό χρόνο. Παρακάτω φαίνονται οι ανταμοιβές του πράκτορα μετά τη φάση της εκπαίδευσης για 100 επεισόδια:

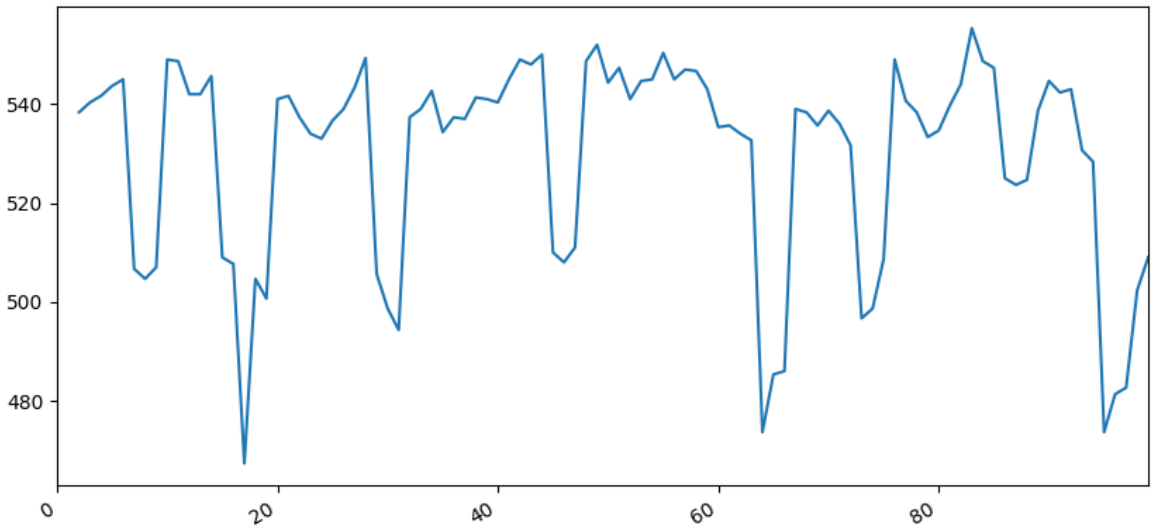


FIGURE 29 ΑΝΤΑΜΟΙΒΕΣ ΣΤΗ ΦΑΣΗ TESTING A3C - ΠΕΡΙΒΑΛΛΟΝ SUPER MARIO 1-1

Όπως μπορούμε να διακρίνουμε ο πράκτορας έχει εκπαιδευτεί και παίρνει ανταμοιβές από το διάστημα 450 έως 550. Σχεδόν σε κάθε περίπτωση τερματίζει την πίστα.

5.5.5 MsPacmanNoFrameskip-v4

Τέλος, ο αλγόριθμος A3C εκπαιδεύτηκε στο περιβάλλον Pacman για 42 χιλιάδες επεισόδια με μεταβλητό ρυθμό μάθησης με αρχική τιμή 0.0003 και γραμμική μείωση στο 0 μετά από 25 εκατομμύρια βήματα. Οι ανταμοιβές του φαίνονται στο παρακάτω διάγραμμα:

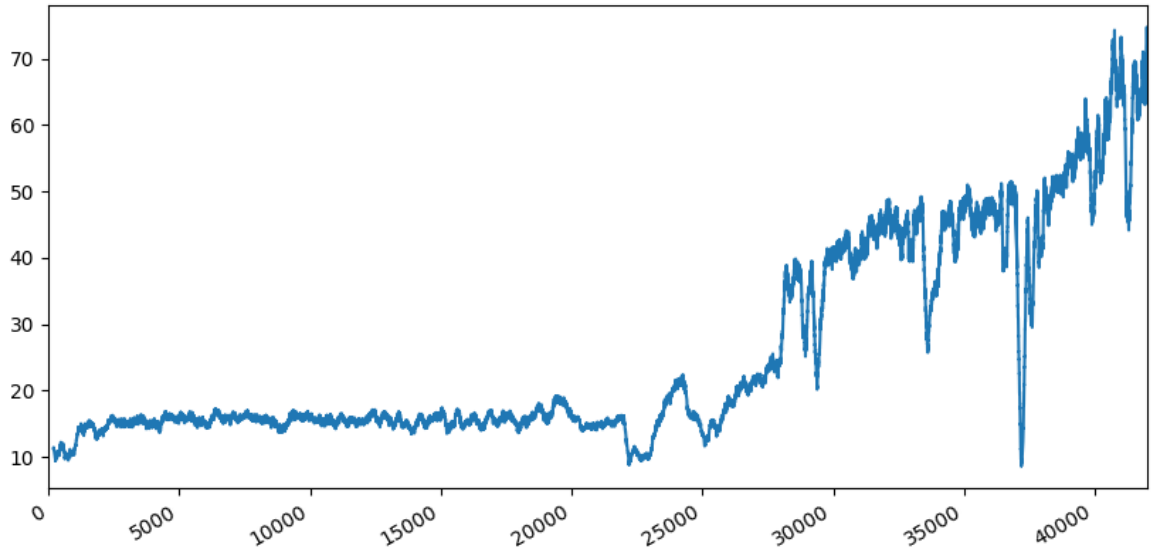


FIGURE 30 ΑΝΤΑΜΟΙΒΕΣ ΣΤΗ ΦΑΣΗ ΤΗΣ ΕΚΠΑΙΔΕΥΣΗΣ A3C - ΠΕΡΙΒΑΛΛΟΝ ATARI MSPACMAN

Όπως μπορούμε να δούμε ο πράκτορας από την αρχική τιμή στο 10 στην αρχή της προπόνησης καταλήγει να μαζεύει 60-80 ανταμοιβή στο τέλος της προπόνησης ενώ αρκετές φορές καταφέρνει να νικήσει και το πρώτο επίπεδο.

Οι ανταμοιβές του στη φάση του testing με τα βάρη που βρήκε για 100 επεισόδια παρουσιάζονται παρακάτω:

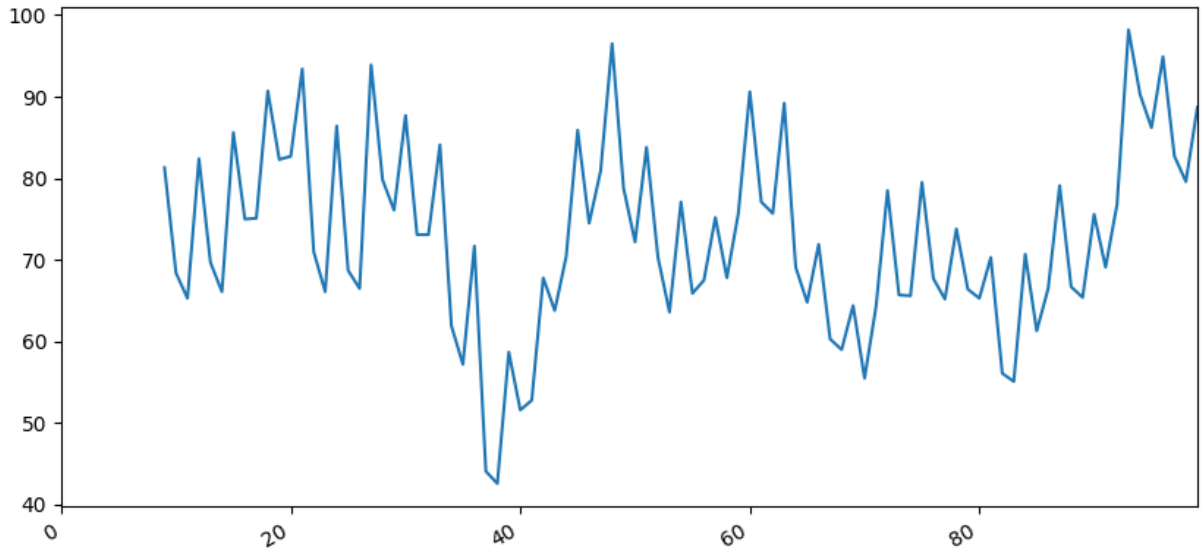


FIGURE 31 FIGURE 28 ΑΝΤΑΜΟΙΒΕΣ ΣΤΗ ΦΑΣΗ TESTING A3C - ΠΕΡΙΒΑΛΛΟΝ MSPACMAN

Και πάλι φαίνεται να παρουσιάζει κάποιες αυξομειώσεις στις ανταμοιβές για τους ίδιους λόγους που αναφέρθηκαν και στον αλγόριθμο double deep q learning. Όμως, φαίνεται να έχει εκπαιδευτεί ώστε να παίρνει ελάχιστη ανταμοιβή 40 και μέγιστη έως και 90-100 μέσο όρο στη διάρκεια των 100 επεισοδίων.

5.6 Συμπεράσματα A3C – Προτάσεις για Βελτίωση

Από τα παραπάνω αποτελέσματα μπορούμε να συμπεράνουμε ότι ο A3C είναι ένας πολύ ισχυρός αλγόριθμος που παράγει αποτελέσματα παρόμοια με αυτά του Deep Q Learning της ενότητας 4. Ένα αρνητικό του A3C που παρατήρησα ήταν ότι είναι πολύ ευαίσθητος στις παραμέτρους του, και ειδικότερα στο ρυθμό μάθησης. Χρειάζεται αρκετό testing για να βρεθούν οι βέλτιστες παράμετροι του συστήματος και μια μικρή απόκλιση από αυτές φέρνει καταστροφικά αποτελέσματα.

Με μια σωστή όμως ρύθμιση των παραμέτρων φαίνεται ότι ο A3C σε πολλά περιβάλλοντα αυξάνει τις ανταμοιβές του πολύ γρήγορα συγκλίνει με μεγάλη ταχύτητα στη βέλτιστη πολιτική. Αυτό οφείλεται κυρίως στην ασύγχρονη φύση του καθώς και σωστή χρήση των μεθόδων κλίσεων πολιτικής και προσέγγισης συνάρτησης ωφέλειας.

Έχουν προταθεί διάφορες βελτιώσεις στον αλγόριθμο A3C. Μια εξ αυτών, ονομαζόμενη Proximal Policy Optimization (PPO), προτείνει την εξόρυξη εμπειριών από τον κάθε Actor για μεγαλύτερο αριθμό βημάτων ($N>128$) και έπειτα η αλλαγή των βαρών του καθολικού δικτύου για K εποχές ($K>3$). Τέλος, οι κλίσεις της πολιτικής περιορίζονται σε ένα διάστημα συνήθως -1.2 έως 1.2 για να αποφευχθεί η απότομη αλλαγή των βαρών και συνεπώς της πολιτικής. Η παραπάνω μέθοδος έχει βρει ιδιαίτερη επιτυχία και πολύ την προτιμούνε από την απλή A3C καθώς με το περιορισμό των κλίσεων της συνάρτησης σφάλματος της πολιτικής γίνεται εφικτό η εκπαίδευση τους για μεγάλο αριθμό εποχών.

6 Επίλογος

Σε αυτή την πτυχιακή εργασία παρουσιάστηκαν δύο διαφορετικές προσέγγισης εκπαίδευσης με τη μέθοδο της ενισχυτικής μάθησης σε μια σειρά από διαφορετικά προβλήματα με διαφορετικούς στόχους. Από τα αποτελέσματα της εκπαίδευσης μπορούμε να συμπεράνουμε ότι και οι δύο αλγόριθμοι παράγουν αξιόλογα αποτελέσματα στα περιβάλλοντα που εξετάστηκαν. Είναι επομένως στην επιλογή του προγραμματιστή για την επιλογή της μεθόδου που θα χρησιμοποιήσει.

Παρακάτω παρουσιάζονται τα πλεονεκτήματα και τα μειονεκτήματα για τους δύο αλγορίθμους εκπαίδευσης ενισχυτικής μάθησης:

Deep Q Learning:

Πλεονεκτήματα:

- *Συγκλίνει στο ολικό ελάχιστο*
- *Μικρή ευαισθησία στις παραμέτρους*
- *Χρήση μόνο δύο δικτύων*

Μειονεκτήματα:

- *Μαθαίνει ντετερμινιστικές πολιτικές*
- *Δεν μπορεί να εφαρμοστεί σε συνεχή περιβάλλοντα ή σε περιβάλλοντα με συνεχής κινήσεις*
- *Στην εκπαίδευση χρειάζεται μεγάλη RAM λόγω της μνήμης εμπειριών*

A3C:

Πλεονεκτήματα

- *Γρήγορη σύγκλιση με σωστή επιλογή παραμέτρων*
- *Μπορεί να μάθει στοχαστικές πολιτικές*
- *Μπορεί να εφαρμοστεί σε συνεχή περιβάλλοντα ή σε συνεχείς ενέργειες*
- *Ενσωματωμένη τάση για εξερεύνηση στην αρχή της εκπαίδευσης*

Αρνητικά

- *Αρκετές φορές συγκλίνει σε τοπικό ελάχιστο*
- *Πολύ ευαίσθητος στις παραμέτρους*

- *Συγκρίσεις μέχρι να βρεθεί σωστός ρυθμός μάθησης*
- *Πολλά δίκτυα στη RAM, 1 καθολικό + ένα για κάθε νήμα*

Όπως φάνηκε από τα παραπάνω συμπεράσματα η ενισχυτική μάθηση είναι ένας πολύ σημαντικός κλάδος της μηχανικής μάθησης, ειδικά όταν χρειάζεται να λυθεί ένας στόχος στον οποίο δεν υπάρχουν έτοιμα δεδομένα εκπαίδευσης και δεν μπορούν να εφαρμοστούν οι υπόλοιπες τεχνικές της μηχανικής μάθησης. Σε αυτή τη περίπτωση όλα τα παραπάνω έδειξαν ότι αρκεί μια συνάρτηση ανταμοιβής και η παρατήρηση του περιβάλλοντος από τον πράκτορα για την εκπαίδευσή του.

Σε αυτή τη πτυχιακή εργασία παρουσιάστηκαν μόνο δύο μέθοδοι, όμως έχουν προταθεί τα τελευταία χρόνια πολλές προσθήκες στη γκάμα των αλγορίθμων της ενισχυτικής μάθησης οι οποίες δίνουν ολοένα και καλύτερα αποτελέσματα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] "Reinforcement Learning: An Introduction- Richard S. Sutton":

http://incompleteideas.net/book/the-book.html

[2] Διαφάνειες από το πανεπιστήμιο UCL στο μάθημα της ενισχυτικής μάθησης καθώς και οι διαλέξεις:

http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

https://www.youtube.com/watch?v=2pWv7GOvuf0&list=PLqYmG7hTraZDM-OYHWgPebj2MfCFzFObQ

[3] Το άρθρο της DeepMind "Playing Atari with Deep Reinforcement Learning":

https://arxiv.org/pdf/1312.5602v1.pdf

[4] Η υλοποίηση του SumTree καθώς και ένα Tutorial στο Deep Q Learning:

https://jaromiru.com/2016/11/07/lets-make-a-dqn-double-learning-and-prioritized-experience-replay/

[5] "Reinforcement Learning in Super Mario Bros" Paper:

http://gabegrand.com/files/super_mario_rl.pdf

[6] "Deep Reinforcement Learning with Double Q-Learning":

https://arxiv.org/pdf/1509.06461.pdf

[7] "Asynchronous Methods for Deep Reinforcement Learning":

https://arxiv.org/pdf/1602.01783.pdf

[8] "Hands on Machine Learning with Scikit-Learn and Tensorflow":

https://www.oreilly.com/library/view/hands-on-machine-learning/9781491962282/

[9] "Simple Reinforcement Learning with Tensorflow: Asynchronous Actor-Critic Agents (A3C)":

https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2

[10] Open AI Gym Enviromets:
https://gym.openai.com/envs/

[11] Tensorflow / Keras Documentation:
https://www.tensorflow.org/
https://keras.io/

[12] “Taking Deep Q Networks a Step further”:
https://sergioskar.github.io/Taking_Deep_Q_Networks_a_step_further/

[13] “Let’s make an A3C Implementation” :
https://jaromiru.com/2017/03/26/lets-make-an-a3c-implementation/

[14] “Trust region and Proximal Policy Optimization”:
https://sergioskar.github.io/TRPO_PPO/

[15] “The idea behind Actor-Critics and how A2C and A3C improve them”:
https://sergioskar.github.io/TRPO_PPO/

[16] “Proximal Policy Optimization Algorithms”:
https://arxiv.org/pdf/1707.06347.pdf

[17] “Machine Learning Course – Coursera”:
https://www.coursera.org/learn/machine-learning

[18] “Deep Q Learning for Video Games – The math of intelligence”:
https://www.youtube.com/watch?v=79pmNdyxEGo

[19] DeepMind Reinforcement Learning:
https://www.youtube.com/watch?v=N0Ld2iTMaMs

[20] Atari Wrappers:
https://github.com/openai/baselines/blob/master/baselines/common/atari_wrappers.py

[21] Playing Tetris with Deep Reinforcement Learning:
http://cs231n.stanford.edu/reports/2016/pdfs/121_Report.pdf

[22] Prioritized Experience Replay:

https://arxiv.org/pdf/1511.05952.pdf

[23] Annealing Variable Implementation (για το ε και το ρυθμό μάθησης):

https://github.com/Kautenja/playing-mario-with-deep-reinforcement-learning/blob/master/src/base/annealing_variable.py

Παράρτημα

Ο κώδικας για τους δύο αλγορίθμους εκπαίδευσης καθώς και τα αποτελέσματα που έδωσαν μπορούν να βρεθούν στο παρακάτω σύνδεσμο:

https://drive.google.com/drive/folders/1qW8b2zzU4IYEYvk05v4XCeyYyyQTvdx7?usp=sharing

Για την εκτέλεση του κώδικα απαιτείται πρώτα η εγκατάσταση της python και των απαραίτητων βιβλιοθηκών. Αυτές περιλαμβάνουν:

- pip install tensorflow
- pip install keras
- pip install gym
- pip install pandas
- pip install gym[atari]
- pip install gym-super-mario-bros
- pip install opencv-python

Κάθε python αρχείο βρίσκεται στο κατάλληλο φάκελο και μπορεί να εκτελεστεί με την εντολή “python ονομα_αρχειου.py”. Σε κάποια μηχανήματα windows βγάζει σφάλματα στην εγκατάσταση της atari βιβλιοθήκης για αυτό προτιμάται η χρήση linux.