Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας

Μέθοδοί Βαθιάς Ενισχυτικής Μάθησης στο Βιντεοπαιχνίδι Supermario Bros

Μελέτη και υλοποίηση

Διπλωματικη Εργασια

του

ΙΩΑΝΝΗ ΒΟΝΔΙΚΑΚΗ

Επιβλέπων: Στέφανος Κόλλιας
Καθηγητής

Αθήνα, Μάιος 2022

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας

Μέθοδοί Βαθιάς Ενισχυτικής Μάθησης στο Βιντεοπαιχνίδι Supermario Bros

Μελέτη και υλοποίηση

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΙΩΑΝΝΗ ΒΟΝΔΙΚΑΚΗ

Επιβλέπων: Στέφανος Κόλλιας
Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 00/00/ 2022.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Στέφανος Κόλλιας	Ανδρέας-Γεώργιος Σταφυλοπάτης	Γιώργος Στάμου
Καθηγητής	Καθηγητής	Καθηγητής

Αθήνα, Μάιος 2022

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφή)

.....
Ιωάννης Βονδικάκης

Μάιος 2022

Περίληψη

Η ενισχυτική μάθηση αφορά έναν πράκτορα που αλληλεπιδρά με το περιβάλλον, μαθαίνοντας μια βέλτιστη πολιτική, μέσω δοκιμής και λάθους, για διαδοχικά προβλήματα λήψης αποφάσεων, σε ένα ευρύ φάσμα πεδίων όπως τις φυσικές επιστήμες, τις κοινωνικές επιστήμες και τη μηχανική. Η βαθιά μάθηση ή αλλιώς τα βαθιά νευρωνικά δίκτυα, έχουν γνωρίσει μεγάλη άνθιση τα τελευταία χρόνια και έχουν επικρατήσει στην ενισχυτική μάθηση. Στόχος της παρούσας διπλωματικής εργασίας είναι η παρουσίαση των βασικών εννοιών της ενισχυτικής μάθησης και η εφαρμογή αλγορίθμων βαθιάς ενισχυτικής μάθησης για την δημιουργία πρακτόρων ικανών να τερματίσουν τα επίπεδά στο βιντεοπαιχνίδι Super Mario Bros. Οι πράκτορες μπορούν να μάθουν διάφορες πολιτικές ελέγχου από ακατέργαστα δεδομένα εικονοστοιχείων με τη χρήση βαθιάς ενισχυτικής μάθησης. Στην εργασία εξετάζονται και συγκρίνονται αλγόριθμοι Q-μάθησης (Q-learning) DQN,DDQN και βελτιστοποίησης πολιτικής (Policy Optimization) PPO, A3C.

Λέξεις Κλειδιά

Τεχνητή Νοημοσύνη, Ενισχυτική Μάθηση, Βαθιά Ενισχυτική Μάθηση, Νευρωνικά Δίκτυα, DQN, DDQN, Policy Gradient, PPO, A3C, Video Games, Super Mario Bros

Abstract

Reinforcement learning involves an agent interacting with the environment, learning an optimal policy, through trial and error, for sequential decision problems, in a wide range of fields such as natural sciences, social sciences and engineering. Deep learning, or deep neural networks, have flourished in recent years and have become prevalent in reinforcement learning. The aim of this thesis is to present the basic concepts of reinforcement learning and to apply deep reinforcement learning algorithms to create agents capable of finishing levels in the video game Super Mario Bros. Agents can learn various control policies from raw pixel data using deep reinforcement learning. The paper examines and compares Q-learning algorithms DQN,DDQN and policy optimization algorithms (Policy Optimization) PPO, A3C.

Keywords

Artificial Intelligence, Reinforcement Learning, Deep Reinforcement Learning, Neural Networks, DQN, DDQN, Policy Gradient, PPO, A3C, Video Games, Super Mario Bros.

στους γονείς μου,
Βασίλη και Μαρία

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον καθηγητή κ. Στέφανο Κόλλια για την επίβλεψη αυτής της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να την εκπονήσω στο εργαστήριο Συστημάτων Τεχνητής Νοημοσύνης και Μάθησης. Επίσης ευχαριστώ ιδιαίτερα την κ. Παρασκευή Τζούβελη για την καθοδήγησή της και την εξαιρετική συνεργασία που είχαμε. Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου για την καθοδήγηση και την ηθική συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια.

Αθήνα, Μάιος 2022

Ιωάννης Βουδικάκης

Περιεχόμενα

Λίστα Σχημάτων

Λίστα Πινάκων

Κεφάλαιο 1

Εισαγωγή

Η παρούσα διπλωματική εργασία πραγματοποιήθηκε στα έτη 2021-2022, στο Εργαστήριο Συστημάτων Τεχνητής Νοημοσύνης και Μάθησης του ΕΜΠ. Το εργαστήριο έχει παρουσιάσει μεγάλη δραστηριότητα στην ανάπτυξη διαφόρων μοντέλων βαθιάς μάθησης. Bayesian μέθοδοί και δίκτυα με capsules έχουν αναπτυχθεί στα [1], [2], [3]. Τεχνικές για προσαρμογή βαθιών νευρωνικών δικτύων σε διάφορα περιβάλλοντα έχουν παρουσιαστεί στα [4], [5]. Τεχνικές για resource allocation σε νευρωνικά δίκτυα ακολουθούν την εργασία [6].

1.1 Μηχανική μάθηση

Η μηχανική μάθηση είναι μια εφαρμογή της τεχνητής νοημοσύνης (AI) που παρέχει σε υπολογιστικά συστήματα την ικανότητα να μαθαίνουν αυτόματα και να βελτιώνονται από την εμπειρία χωρίς να προγραμματίζονται ρητά. Η μηχανική μάθηση επικεντρώνεται στην ανάπτυξη προγραμμάτων που μπορούν να έχουν πρόσβαση σε δεδομένα και να τα χρησιμοποιούν για να μαθαίνουν μόνα τους. Αντιγράφοντας τον τρόπο με τον οποίο μαθαίνει ο άνθρωπος, βελτιώνουν σταδιακά την ακρίβειά τους. [7], [8]

Η διαδικασία της μάθησης ξεκινά με παρατηρήσεις ή δεδομένα, όπως παραδείγματα, άμεση εμπειρία ή οδηγίες, προκειμένου να αναζητηθούν μοτίβα στα δεδομένα και να ληφθούν καλύτερες αποφάσεις στο μέλλον με βάση τα παραδείγματα που παρέχουμε. Ο πρωταρχικός στόχος είναι να επιτραπεί στους υπολογιστές να μαθαίνουν αυτόματα χωρίς ανθρώπινη παρέμβαση ή βοήθεια και να προσαρμόζουν τις ενέργειες τους ανάλογα. Η μηχανική μάθηση επιτρέπει την ανάλυση τεράστιων ποσοτήτων δεδομένων, ενώ γενικά παρέχει ταχύτερα και ακριβέστερα αποτελέσματα από άλλες μεθόδους. Μπορεί επίσης να απαιτεί πρόσθετο χρόνο και πόρους για τη σωστή εκπαίδευσή της. [9],

1.2 Μέθοδοι μηχανικής μάθησης

Η κλασική μηχανική μάθηση συχνά κατηγοριοποιείται με βάση τον τρόπο με τον οποίο ένας αλγόριθμος μαθαίνει να γίνεται πιο ακριβής στις προβλέψεις του. Υπάρχουν τέσσερις βασικές προσεγγίσεις: επιβλεπόμενη μάθηση , μη επιβλεπόμενη μάθηση, ημι-επιβλεπόμενη μάθηση και ενισχυτική μάθησή. Ο τύπος αλγορίθμου που επιλέγεται εξαρτάται από τον τύπο των δεδομένων και το είδος του προβλήματος που καλούμαστε να αντιμετωπίσουμε. Στην παρούσα εργασία θα επικεντρωθούμε στην ενισχυτική μάθησή. [8]

1.2.1 Ενισχυτική μάθηση

Οι αλγόριθμοι ενισχυτικής μάθησης αλληλεπιδρούν με το περιβάλλον τους. Η ενισχυτική μάθηση λειτουργεί με τον προγραμματισμό ενός αλγορίθμου με έναν διακριτό στόχο και ένα καθορισμένο σύνολο κανόνων για την επίτευξη αυτού του στόχου. Να επιδιώκει δηλαδή ο αλγόριθμος θετικές ανταμοιβές, τις οποίες λαμβάνει όταν εκτελεί μια ενέργεια που είναι επωφελής ως προς τον τελικό στόχο, και να αποφεύγει αρνητικές ανταμοιβές, τις οποίες λαμβάνει όταν εκτελεί μια ενέργεια που τον απομακρύνει από τον τελικό στόχο. Αυτή η μέθοδος επιτρέπει στις μηχανές και στους πράκτορες να καθορίζουν αυτόματα την ιδανική συμπεριφορά σε ένα συγκεκριμένο πλαίσιο, προκειμένου να μεγιστοποιήσουν την απόδοσή τους. Απαιτείται ανατροφοδότηση για να μάθει ο πράκτορας ποια ενέργεια είναι η καλύτερη. Αυτό είναι γνωστό ως σήμα ενίσχυσης. [10]

Η ενισχυτική μάθηση χρησιμοποιείται συχνά σε τομείς όπως:

- Ρομποτική: Τα ρομπότ μπορούν να μάθουν να εκτελούν εργασίες στον φυσικό κόσμο.
- Βιντεοπαιχνίδια: Η μάθηση ενίσχυσης έχει χρησιμοποιηθεί για να μάθουν ρομπότ να παίζουν διάφορα βιντεοπαιχνίδια.
- Διαχείριση πόρων: Δεδομένων πεπερασμένων πόρων και ενός καθορισμένου στόχου, η ενισχυτική μάθηση μπορεί να βοηθήσει τις επιχειρήσεις να σχεδιάσουν τον τρόπο κατανομής των πόρων.

1.3 Ιστορία της ενισχυτικής μάθησης

Καταβολές στη μάθηση των ζώων

Η ενισχυτική μάθηση προέρχεται από τη μάθηση των ζώων στην πειραματική ψυχολογία και τη θεωρία του βέλτιστου ελέγχου, ενώ παράλληλα αντλεί ιδέες από τη νευροεπιστήμη.

Ξεκινώντας από τη ζωική μάθηση, ο Edward Thorndike περιέγραψε την ουσία της μάθησης δοκιμής-λάθους με τον "Νόμο του Αποτελέσματος" [11] το 1911, ο οποίος, για να τον παραφράσουμε, δηλώνει ότι ένα ζώο θα επιδιώξει την επανάληψη ενεργειών αν αυτές ενισχύουν την ικανοποίηση και θα αποτραπεί από ενέργειες που προκαλούν δυσφορία. Επιπλέον, όσο μεγαλύτερο είναι το επίπεδο της ευχαρίστησης ή του πόνου, τόσο μεγαλύτερη είναι η επιδίωξη ή η αποτροπή από την ενέργεια. Ο Νόμος του Αποτελέσματος περιγράφει το αποτέλεσμα της ενίσχυσης της συμπεριφοράς από θετικά ερεθίσματα και θεωρείται ευρέως ως η βασική αρχή καθορισμού των μελλοντικών περιγραφών της συμπεριφοράς. Ο Νόμος του Αποτελέσματος συνδυάζει την επιλεκτική και τη συνειρμική μάθηση, όπου η επιλεκτική περιλαμβάνει τη δοκιμή εναλλακτικών επιλογών και την επιλογή από αυτές με βάση τα αποτελέσματα, ενώ η συνειρμική είναι όπου οι επιλογές βρίσκονται με επιλογή και συνδέονται με συγκεκριμένες καταστάσεις. Ο όρος ενίσχυση χρησιμοποιήθηκε επίσημα στο πλαίσιο της μάθησης των ζώων το 1927 από τον Pavlov, ο οποίος περιέγραψε την ενίσχυση ως την ενίσχυση ενός προτύπου συμπεριφοράς που οφείλεται στο ότι ένα ζώο λαμβάνει ένα

ερέθισμα - έναν ενισχυτή - σε μια χρονικά εξαρτώμενη σχέση με ένα άλλο ερέθισμα ή με μια απόκριση.

Οι ανοργάνωτες μηχανές του Turing

Το 1948, ο Άλαν Τούρινγκ παρουσίασε μια οραματική επισκόπηση της προοπτικής κατασκευής μηχανών ικανών για ευφυή συμπεριφορά σε μια έκθεση με τίτλο "Intelligent Machinery" [12]. Ο Τιούρινγκ ήταν ίσως ο πρώτος που πρότεινε τη χρήση τυχαία συνδεδεμένων δικτύων κόμβων που μοιάζουν με νευρώνες για την εκτέλεση υπολογισμών και πρότεινε την κατασκευή μεγάλων, εγκεφαλικών δικτύων τέτοιων νευρώνων ικανών να εκπαιδευτούν όπως θα δίδασκε κανείς ένα παιδί. Ο Τούρινγκ ονόμασε τα δίκτυά του "μη οργανωμένες μηχανές".

Ο Τούρινγκ περιέγραψε τρεις τύπους μη οργανωμένων μηχανών. Οι ανοργάνωτες μηχανές τύπου A και B αποτελούνται από τυχαία συνδεδεμένους νευρώνες δύο καταστάσεων. Οι ανοργάνωτες μηχανές τύπου P, οι οποίες δεν μοιάζουν με νευρώνες, έχουν "μόνο δύο παρεμβαλλόμενες εισόδους, μία για ευχαρίστηση ή ανταμοιβή και μία για πόνο ή τιμωρία". Ο Τούρινγκ μελέτησε τις μηχανές τύπου P για να προσπαθήσει να ανακαλύψει διαδικασίες εκπαίδευσης ανάλογες με αυτές που μαθαίνουν τα παιδιά. Δήλωσε ότι με την εφαρμογή "κατάλληλων συμπερασμάτων, που μιμούνται την εκπαίδευση", μια μηχανή τύπου B μπορεί να εκπαιδευτεί ώστε "να κάνει οποιαδήποτε απαιτούμενη εργασία, δεδομένου επαρκούς χρόνου και υπό την προϋπόθεση ότι ο αριθμός των μονάδων είναι επαρκής".

Η εκμάθηση μέσω δοκιμής και λάθους οδήγησε στην παραγωγή πολλών ηλεκτρομηχανικών μηχανών. Ο Thomas Ross, το 1933 κατασκεύασε μια μηχανή που μπορούσε να βρει το δρόμο της μέσα από έναν απλό λαβύρινθο και να θυμάται τη διαδρομή μέσω της διαμόρφωσης των διακοπτών. Το 1952, ο Κλοντ Σάνον παρουσίασε ένα ποντίκι που έτρεχε σε λαβύρινθο και ονομαζόταν Θησέας, το οποίο χρησιμοποιούσε τη μέθοδο δοκιμής και σφάλματος για να πλοηγηθεί σε έναν λαβύρινθο. Το 1954, ο Μάρβιν Μίνσκι συζήτησε υπολογιστικές μεθόδους ενισχυτικής μάθησης και περιέγραψε την κατασκευή μιας αναλογικής μηχανής που αποτελούνταν από εξαρτήματα τα οποία ονόμασε SNARCs (Stochastic Neural-Analog Reinforcement Calculators). Οι SNARCs προορίζονταν να μοιάζουν με τις τροποποιήσιμες συναπτικές συνδέσεις στον εγκέφαλο.

Η έρευνα στις υπολογιστικές διαδικασίες δοκιμής και σφάλματος γενικεύτηκε τελικά στην αναγνώριση προτύπων προτού απορροφηθεί στην επιβλεπόμενη μάθηση, όπου οι πληροφορίες σφάλματος χρησιμοποιούνται για την ενημέρωση των βαρών των συνδέσεων των νευρώνων. Η έρευνα στην RL εξασθένησε κατά τη διάρκεια της δεκαετίας του 1960 και του 1970. Ωστόσο, το 1963, αν και σχετικά άγνωστος, ο John Andreae ανέπτυξε πρωτοποριακή έρευνα, συμπεριλαμβανομένου του συστήματος STELLA, το οποίο μαθαίνει μέσω της αλληλεπίδρασης με το περιβάλλον του, και μηχανές με "εσωτερικό μονόλογο" και αργότερα μηχανές που μπορούν να μάθουν από έναν δάσκαλο. [13]

Καταβολές στον Βέλτιστο Έλεγχο

Τη δεκαετία του 1950 ξεκίνησε η έρευνα για τον Βέλτιστο Έλεγχο ως ένα επίσημο πλαίσιο για τον ορισμό μεθόδων βελτιστοποίησης για την εξαγωγή πολιτικών ελέγχου σε προβλήματα ελέγχου συνεχούς χρόνου, όπως έδειξαν οι Pontryagin και Neustadt το 1962. Ο Richard

Bellman ανέπτυξε τον δυναμικό προγραμματισμό ως μέθοδο μαθηματικής βελτιστοποίησης και προγραμματισμού σε υπολογιστή για την επίλυση προβλημάτων ελέγχου. Η διαδικασία ορίζει μια συναρτησιακή εξίσωση χρησιμοποιώντας την κατάσταση του δυναμικού συστήματος και επιστρέφει αυτό που αναφέρεται ως συνάρτηση βέλτιστης τιμής. Η βέλτιστη συνάρτηση αναφέρεται συνήθως ως εξίσωση Bellman. Ο Bellman εισήγαγε τη Μαρκοβιανή Διαδικασία Αποφάσεων (MDP), την οποία ορίζουμε ως μια διακριτή στοχαστική εκδοχή του προβλήματος βέλτιστου ελέγχου. Ο Ronald Howard, το 1960 επινόησε τη μέθοδο επανάληψης πολιτικής για τις MDPs. [14] Όλα αυτά αποτελούν βασικά στοιχεία που στηρίζουν τη θεωρία και τους αλγορίθμους της σύγχρονης ενισχυτικής μάθησης.

Αυτόματα μάθησης (Learning Automata)

Στις αρχές της δεκαετίας του 1960 ξεκίνησε η έρευνα στα αυτοματοποιημένα συστήματα μάθησης, η οποία ανάγεται στον Michael Lvovitch Tsetlin στη Σοβιετική Ένωση. Ένα αυτόματο μάθησης είναι μια προσαρμοστική μονάδα λήψης αποφάσεων που βρίσκεται σε ένα τυχαίο περιβάλλον και μαθαίνει τη βέλτιστη δράση μέσω επαναλαμβανόμενων αλληλεπιδράσεων με το περιβάλλον της. Τα βήματα επιλέγονται σύμφωνα με μια συγκεκριμένη κατανομή πιθανοτήτων, με βάση την απόκριση από το περιβάλλον. Τα αυτόματα μάθησης θεωρούνται ως επαναλήπτες πολιτικής στην RL. Ο Tsetlin επινόησε το Αυτόματο Tsetlin, το οποίο θεωρείται ακόμη πιο θεμελιώδης και ευέλικτος μηχανισμός μάθησης από τον τεχνητό νευρώνα. Το Αυτόματο Tsetlin είναι μια από τις πρωτοποριακές λύσεις στο γνωστό πρόβλημα της πολυόπλων ληστών και συνεχίζει να χρησιμοποιείται για την ταξινόμηση προτύπων και αποτέλεσε τον πυρήνα πιο προηγμένων σχεδίων αυτομάτων μάθησης, συμπεριλαμβανομένου του αποκεντρωμένου ελέγχου και της ισοκατανομής και της ελαττωματικής διχοτομικής αναζήτησης.

Ηδονιστικοί νευρώνες

Στα τέλη της δεκαετίας του 1970 και στις αρχές της δεκαετίας του 1980, ο Harry Klopf υποστήριξε ότι τα συστήματα που προσπαθούν να μεγιστοποιήσουν μια ποσότητα είναι ποιοτικά διαφορετικά από τα συστήματα που αναζητούν ισορροπία. Επιπλέον, υποστήριξε ότι τα συστήματα μεγιστοποίησης είναι αναπόσπαστο στοιχείο για την κατανόηση κρίσιμων πτυχών της φυσικής νοημοσύνης και την κατασκευή τεχνητής νοημοσύνης. Ο Klopf υπέθεσε ότι οι νευρώνες είναι ατομικά ηδονιστικοί, δεδομένου ότι εργάζονται για να μεγιστοποιήσουν ένα νευρωνικό-τοπικό ανάλογο της ευχαρίστησης ενώ ελαχιστοποιούν ένα νευρωνικό-τοπικό του πόνου.

Η ιδέα του Klopf για τους ηδονιστικούς νευρώνες ήταν ότι οι νευρώνες εφαρμόζουν μια τοπική εκδοχή του νόμου του αποτελέσματος. Υπέθεσε ότι τα συναπτικά βάρη των νευρώνων αλλάζουν με την εμπειρία. Όταν ένας νευρώνας πυροδοτεί ένα δυναμικό δράσης, όλες οι συνάψεις που συμβάλλουν στο δυναμικό δράσης αλλάζουν τις αποδόσεις. Εάν το δυναμικό δράσης ανταμείβεται, η αποτελεσματικότητα όλων των επιλέξιμων συνάψεων αυξάνεται (ή μειώνεται εάν τιμωρείται). Επομένως, οι συνάψεις αλλάζουν για να αλλάξουν τα μοτίβα πυροδότησης του νευρώνα ώστε να αυξηθεί η πιθανότητα ο νευρώνας να ανταμειφθεί και να μειωθεί η πιθανότητα να τιμωρηθεί από το περιβάλλον του.

Αυτή η υπόθεση παράγει μια σημαντική διάκριση μεταξύ της μάθησης με επίβλεψη, η οποία είναι ουσιαστικά μια διαδικασία αναζήτησης ισορροπίας, και της ενισχυτικής μάθησης, η οποία είναι ουσιαστικά ένα σύστημα που καθοδηγείται από την αξιολόγηση, όπου οι αποφάσεις του μαθητή εξελίσσονται ως απάντηση στις εμπειρίες του. Τόσο η διόρθωση σφαλμάτων όσο και η RL είναι διαδικασίες βελτιστοποίησης, αλλά η διόρθωση σφαλμάτων είναι πιο περιορισμένη, ενώ η RL είναι γενικευμένη και παρακινείται από τη μεγιστοποίηση των ανταμοιβών μέσω της βελτιστοποίησης της δράσης.

Χρονική διαφορά

Η εκμάθηση της χρονικής διαφοράς (TD) εμπνέεται από τη μαθηματική διαφοροποίηση και αποσκοπεί στη δημιουργία ακριβών προβλέψεων ανταμοιβής από καθυστερημένες ανταμοιβές. Η TD προσπαθεί να προβλέψει το συνδυασμό της άμεσης ανταμοιβής και της πρόβλεψης της ανταμοιβής της στο επόμενο χρονικό βήμα. Όταν φτάσει το επόμενο χρονικό βήμα, η τελευταία πρόβλεψη συγκρίνεται με το τι αναμενόταν να γίνει με νέες πληροφορίες. Εάν υπάρχει διαφορά, ο αλγόριθμος υπολογίζει το σφάλμα, το οποίο είναι η "χρονική διαφορά" για να προσαρμόσει την παλιά πρόβλεψη προς την τελευταία πρόβλεψη. Ο αλγόριθμος έχει ως στόχο να φέρνει την παλιά και τη νέα πρόβλεψη πιο κοντά σε κάθε χρονικό βήμα, διασφαλίζοντας ότι ολόκληρη η αλυσίδα των προβλέψεων γίνεται σταδιακά πιο ακριβής.

Η μάθηση TD συνδέεται στενότερα με τον Sutton, του οποίου η διδακτορική διατριβή του 1984 ασχολήθηκε με τη μάθηση TD και του οποίου η εργασία του 1988, στην οποία χρησιμοποιήθηκε για πρώτη φορά ο όρος "χρονική διαφορά", έχει γίνει η καθοριστική αναφορά. [15]

Η προέλευση των μεθόδων της χρονικής διαφοράς έχει ως ισχυρό κίνητρο τις θεωρίες της ζωικής μάθησης, ιδιαίτερα την έννοια των δευτερογενών ενισχυτών. Ένας δευτερεύων ενισχυτής είναι ένα ερέθισμα που συνδυάζεται με έναν πρωτεύοντα ενισχυτή, για παράδειγμα την παρουσία τροφής. Ο δευτερεύων ενισχυτής υιοθετεί παρόμοιες ιδιότητες με τον πρωτεύοντα. Η μέθοδος της χρονικής διαφοράς διαπλέκεται με τη μέθοδο δοκιμής και σφάλματος όταν ο Klopf το 1975 [16] διερεύνησε τη μάθηση ενίσχυσης σε μεγάλα συστήματα, όπως αποσυντίθεται σε επιμέρους υποσυστήματα της πιο εκτεταμένης διαδικασίας, καθένα από τα οποία έχει τις διεγερτικές εισροές του ως ανταμοιβές και τις ανασταλτικές εισροές του ως τιμωρίες, και κάθε ένα μπορεί να ενισχύει το άλλο.

Η ενσωμάτωση της θεωρίας της ζωικής μάθησης με μεθόδους μάθησης που καθοδηγούνται από αλλαγές σε διαχρονικά διαδοχικές προβλέψεις, συμπεριλαμβανομένου του προβλήματος της χρονικής ανάθεσης πίστωσης, οδήγησε σε μια έκρηξη στην έρευνα της ενισχυτικής μάθησης. Ειδικότερα, η ανάπτυξη της "Αρχιτεκτονικής του Δράστη-Κριτικού", όπως εφαρμόστηκε στο πρόβλημα της εξισορρόπησης πόλων από τους Barto et al. το 1983. Οι μέθοδοι Actor-critic είναι μέθοδοι TD με ξεχωριστή δομή μνήμης για τη ρητή αναπαράσταση της πολιτικής ανεξάρτητα από τη συνάρτηση αξίας. Η δομή πολιτικής, η οποία χρησιμοποιείται για την επιλογή ενεργειών, είναι γνωστή ως δράστης, και η εκτιμώμενη συνάρτηση αξίας, η οποία επικρίνει τις ενέργειες που γίνονται από τον δράστη, είναι γνωστή ως κριτικός. Η κριτική παίρνει τη μορφή ενός σφάλματος TD, το οποίο αποτελεί τη μοναδική έξοδο του κριτικού και οδηγεί όλη τη μάθηση τόσο στον δράστη όσο και στον κριτικό. Το 1984 και το

1986, η αρχιτεκτονική Actor-Critic επεκτάθηκε για να ενσωματωθεί με τεχνικές νευρωνικών δικτύων οπισθοδιάδοσης.

Το 1992, ο Gerry Tesauro ανέπτυξε ένα πρόγραμμα που απαιτούσε ελάχιστες γνώσεις τάβλι, αλλά μάθαινε να παίζει το παιχνίδι σε επίπεδο grandmaster. Ο αλγόριθμος εκμάθησης συνδύαζε τον αλγόριθμο TD-lambda και μια μη γραμμική προσέγγιση συνάρτησης χρησιμοποιώντας ένα πολυστρωματικό νευρωνικό δίκτυο εκπαιδευμένο με οπισθοδιάδοση σφαλμάτων TD. Με βάση την επιτυχία του TD-Gammon και την περαιτέρω ανάλυση, οι καλύτεροι ανθρώπινοι παίκτες παίζουν τώρα τις αντισυμβατικές θέσεις ανοίγματος που έμαθε ο αλγόριθμος.

Q-Learning

Ο Chris Watkins εισήγαγε την Q-learning το 1989 στη διδακτορική του διατριβή με τίτλο "Learning from Delayed Rewards", η οποία εισήγαγε ένα μοντέλο ενισχυτικής μάθησης ως σταδιακά βελτιστοποιημένου ελέγχου μιας Μαρκοβιανής Διαδικασίας Απόφασης και πρότεινε την Q-learning ως έναν τρόπο εκμάθησης του βέλτιστου ελέγχου απευθείας χωρίς να μοντελοποιούνται οι πιθανότητες μετάβασης ή οι αναμενόμενες ανταμοιβές της Μαρκοβιανής Διαδικασίας Απόφασης. Οι Watkins και Peter Dayan παρουσίασαν μια απόδειξη σύγκλισης το 1992. Μια συνάρτηση Q-value μας δείχνει πόσο καλή είναι μια συγκεκριμένη ενέργεια, δεδομένης μιας κατάστασης για έναν πράκτορα που ακολουθεί μια πολιτική. Η εκμάθηση Q είναι η διαδικασία επαναληπτικής ενημέρωσης των τιμών Q για κάθε ζεύγος κατάστασης-δράσης χρησιμοποιώντας την εξίσωση Bellman μέχρι η συνάρτηση Q να συγκλίνει τελικά στο Q*. Η Q-μάθηση είναι ένας αλγόριθμος ενισχυτικής μάθησης χωρίς μοντέλα και μπορεί να χειριστεί στοχαστικές μεταβάσεις και ανταμοιβές χωρίς προσαρμογές.

1.4 Ιστορία της βαθιάς ενισχυτικής μάθησης

Παράλληλα με το αυξανόμενο ενδιαφέρον για τα νευρωνικά δίκτυα που ξεκίνησε στα μέσα της δεκαετίας του 1980, το ενδιαφέρον αυξήθηκε και για τη βαθιά ενισχυτική μάθηση, όπου ένα νευρωνικό δίκτυο αναπαριστά πολιτικές ή συναρτήσεις αξίας. Το TD-Gammon ήταν η πρώτη επιτυχής εφαρμογή της ενισχυτικής μάθησης με νευρωνικά δίκτυα. Γύρω στο 2012, μια επανάσταση στη βαθιά μάθηση προκλήθηκε από τις γρήγορες υλοποιήσεις των νευρωνικών δικτύων συνελίξεων σε μονάδες γραφικής επεξεργασίας για την όραση υπολογιστών, γεγονός που οδήγησε σε αυξημένο ενδιαφέρον για τη χρήση βαθιών νευρωνικών δικτύων ως προσεγγίσεις συναρτήσεων σε διάφορους τομείς. Η εφαρμογή νευρωνικών δικτύων είναι ιδιαίτερα χρήσιμη για την αντικατάσταση αλγορίθμων επανάληψης τιμών που ενημερώνουν απευθείας τους πίνακες q-τιμών καθώς ο πράκτορας μαθαίνει. Η επανάληψη τιμών είναι κατάλληλη για εργασίες με μικρό χώρο καταστάσεων, αλλά αν υπάρχουν πιο σύνθετα περιβάλλοντα, ο αριθμός των υπολογιστικών πόρων και ο χρόνος που απαιτούνται για τη διέλευση της νέας κατάστασης και την τροποποίηση των τιμών q θα είναι εξαιρετικά απαγορευτικός ή ανέφικτος. Αντί του άμεσου υπολογισμού των τιμών Q μέσω επαναλήψεων τιμών, μια προσέγγιση συναρτήσεων μπορεί να εκτιμήσει τη βέλτιστη συνάρτηση Q. Ένα νευρωνικό δίκτυο λαμβάνει καταστάσεις από ένα περιβάλλον ως είσοδο και εξάγει εκτιμώμενες τιμές

Q για κάθε ενέργεια που μπορεί να επιλέξει ένας πράκτορας σε αυτές τις καταστάσεις. Οι τιμές συγκρίνονται με τις τιμές-στόχους Q* για τον υπολογισμό της απώλειας. Τα βάρη του νευρωνικού δικτύου ενημερώνονται χρησιμοποιώντας οπισθοδιάδοση και στοχαστική κάθοδο κλίσης για να παράγουν τιμές Q που ελαχιστοποιούν το σφάλμα και συγκλίνουν στις βέλτιστες ενέργειες του πράκτορα.

Google DeepMind

Γύρω στο 2013, η DeepMind ανέπτυξε τη βαθιά Q-learning, έναν συνδυασμό της αρχιτεκτονικής του νευρωνικού δικτύου συνέλιξης και της Q-learning. Η βαθιά Q-learning διευκολύνει την αναπαραγωγή εμπειριών, η οποία αποθηκεύει και αναπαράγει καταστάσεις και επιτρέπει στο δίκτυο να μαθαίνει σε μικρές παρτίδες για να αποφεύγεται η στρέβλωση της εκπαίδευσης και να επιταχύνεται η υλοποίηση. Δοκίμασαν το σύστημα σε βιντεοπαιχνίδια όπως το Space Invaders και το Breakout. Χωρίς τροποποίηση του κώδικα, το δίκτυο μαθαίνει πώς να παίζει το παιχνίδι και, μετά από αρκετές επαναλήψεις, ξεπερνά την ανθρώπινη απόδοση. Η DeepMind δημοσίευσε περαιτέρω έρευνα σχετικά με το σύστημά τους που ξεπερνά τις ανθρώπινες ικανότητες σε άλλα παιχνίδια όπως το Seaquest και το Q*Bert.

Το 2014, η DeepMind δημοσίευσε έρευνα σχετικά με το πρόγραμμα υπολογιστή που είναι σε θέση να παίξει Go. Τον Οκτώβριο του 2015, ένα υπολογιστικό πρόγραμμα Go που ονομάζεται AlphaGo [17] νίκησε τον πρωταθλητή Ευρώπης στο Go, Fan Hui. Το γεγονός αυτό ήταν η πρώτη φορά που η τεχνητή νοημοσύνη νίκησε έναν επαγγελματία παίκτη του Γκο. Τον Μάρτιο του 2016, το AlphaGo νίκησε τον Lee Sedol, έναν από τους πιο υψηλόβαθμους παίκτες στον κόσμο, με σκορ 4-1 σε έναν αγώνα πέντε παιχνιδιών. Στο Future of Go Summit του 2017, ο AlphaGo κέρδισε σε έναν αγώνα τριών παιχνιδιών τον Ke Jie, ο οποίος ήταν ο νούμερο ένα παίκτης στην παγκόσμια κατάταξη για δύο χρόνια. Αργότερα την ίδια χρονιά, μια βελτιωμένη έκδοση του AlphaGo, το AlphaGo Zero, νίκησε το AlphaGo 100 παιχνίδια με 0. Αυτή η νέα έκδοση νίκησε τον προκάτοχό της μετά από τρεις ημέρες με λιγότερη επεξεργαστική ισχύ από το AlphaGo, το οποίο συγκριτικά χρειάστηκε μήνες για να μάθει να παίζει.

Στην εργασία της Google με το AlphaZero το 2017, το σύστημα μπόρεσε να παίξει σκάκι σε υπεράνθρωπο επίπεδο μέσα σε τέσσερις ώρες εκπαίδευσης, χρησιμοποιώντας 5.000 μονάδες επεξεργασίας τανυστών πρώτης γενιάς και 64 μονάδες επεξεργασίας τανυστών δεύτερης γενιάς.

Σύγχρονες εξελίξεις

Η ερευνητική κοινότητα βρίσκεται ακόμη σε πρώιμο στάδιο για να κατανοήσει σε βάθος πόσο πρακτική είναι η βαθιά ενισχυτική μάθηση σε άλλους τομείς. Το AlphaFold, που αναπτύχθηκε από την DeepMind, εφαρμόζει τεχνητή νοημοσύνη στην αναδίπλωση αμινοξέων, έναν από τους σημαντικότερους στόχους που επιδιώκει η υπολογιστική βιολογία. Είναι απαραίτητος για φαρμακευτικές εφαρμογές όπως ο σχεδιασμός φαρμάκων και για την βιοτεχνολογία π.χ ο σχεδιασμός νέων ενζύμων. Η βαθιά ενισχυτική μάθηση έχει επιδείξει εξαιρετική επάρκεια στην επίλυση προβλημάτων μέσα σε περιορισμένα περιβάλλοντα. Πιθανές εφαρμογές στην πραγματική ζωή περιλαμβάνουν τη ρομποτική, την επεξεργασία δομημένων

ιατρικών εικόνων, τα αυτοκινούμενα αυτοκίνητα.

Έχουν σημειωθεί εξελίξεις για να καταστεί η βαθιά ενισχυτική μάθηση πιο αποτελεσματική. Το Google Brain πρότεινε το Adaptive Behavior Policy Sharing,[18] μια στρατηγική βελτιστοποίησης που επιτρέπει την επιλεκτική ανταλλαγή πληροφοριών σε μια ομάδα πρακτόρων. Η DeepMind δημοσίευσε έρευνα το 2020, διερευνώντας τη στρατηγική Never Give Up, [19] η οποία χρησιμοποιεί τους k-κοντινότερους γείτονες πάνω στην πρόσφατη εμπειρία του πράκτορα για την εκπαίδευση των κατευθυνόμενων διερευνητικών πολιτικών για την επίλυση σύνθετων παιχνιδιών εξερεύνησης.

1.5 Παιχνίδια και τεχνητή νοημοσύνη

Τα παιχνίδια έχουν μακρά ιστορία στην εξέλιξη της τεχνητής νοημοσύνης και της μηχανικής μάθησης. Η κατασκευή συνθετικών κόσμων στα παιχνίδια έχει χρησιμεύσει ως ένα χρήσιμο πεδίο δοκιμών για αλγορίθμους τεχνητής νοημοσύνης. Επιπλέον, η εφαρμογή της τεχνητής νοημοσύνης σε παιχνίδια επιτρέπει την κατασκευή ανταγωνιστικών αντιπάλων για την πρόκληση των παικτών.

Η τεχνητή νοημοσύνη έχει εφαρμοστεί σε παιχνίδια από τις απαρχές της, σε παραδοσιακά παιχνίδια όπως το σκάκι μέχρι σύγχρονα παιχνίδια στρατηγικής πραγματικού χρόνου όπως το StarCraft II. Μια από τις πρώτες εφαρμογές της παραδοσιακής μηχανικής μάθησης σε παιχνίδια εφαρμόστηκε από τον Arthur Samuel (ο οποίος επινόησε τον όρο "μηχανική μάθηση"), το 1956, στο παιχνίδι της νταμας. Το 1996, το Deep Blue της IBM νίκησε τον παγκόσμιο πρωταθλητή στο σκάκι Γκάρι Κασπάροφ, ενώ το 2017, το AlphaGo της Google χρησιμοποιώντας βαθιά μάθηση και αναζήτηση Μόντε Κάρλο για την επιλογή των κινήσεων, νίκησε τον No.1 παίκτη Go στην παγκόσμια κατάταξη. Τέλος η OpenAI ανέπτυξε το OpenAI Five για το Dota 2 που κατάφερε να νικήσει τους παγκόσμιους πρωταθλητές τον Απρίλιο του 2019. Η τεχνητή νοημοσύνη δημιουργήθηκε από το μηδέν και δεν περιλάμβανε καμία εγγενή γνώση του παιχνιδιού ή ευρετικές λειτουργίες που να καθοδηγούν το παιχνίδι της. Έμαθε να παίζει το παιχνίδι παίζοντας με τον εαυτό της, ξεκινώντας με τυχαία βάρη για τα νευρωνικά της δίκτυα. [20]

Στην παρούσα διπλωματική εργασία, γίνεται προσπάθεια να παίξουμε Super Mario Bros χρησιμοποιώντας πράκτορες βαθιάς ενισχυτικής μάθησης και να συγκρίνουμε την απόδοσή τους. Το Super Mario Bros είναι ένα παιχνίδι που αποτελείται από δισδιάστατα επίπεδα όπου ο παίκτης αναλαμβάνει τον έλεγχο του Mario και προσπαθεί να πλοηγηθεί προς τον ιστό της σημαίας που βρίσκεται στην άκρη της δεξιάς πλευράς κάθε επιπέδου, και σηματοδοτεί το στόχο του επιπέδου.

1.6 Οργάνωση του τόμου

Η εργασία αυτή είναι οργανωμένη σε έξι κεφάλαια: Στο Κεφάλαιο 2 γίνεται εμβάθυνση στις έννοιές της ενισχυτικής μάθησης όπως οι μαρκοβιανες διαδικασίες αποφάσεων ,ο δυναμικός προγραμματισμός και η ενισχυτική μάθηση χωρίς μοντέλα. Στο Κεφάλαιο 4 παρουσιάζονται οι αλγόριθμοι βαθιάς ενισχυτικής μάθησης που θα χρησιμοποιήσουμε στο πειραματικό

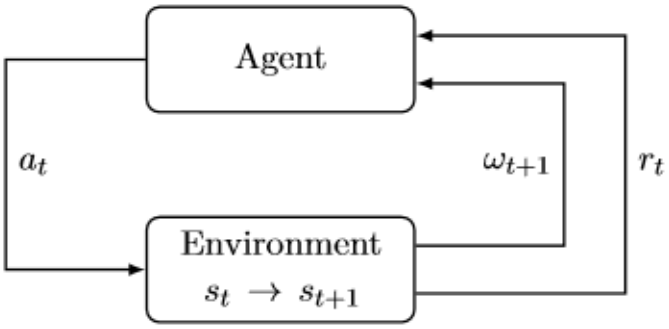
κομμάτι της εργασίας. Στο Κεφάλαιο 5 περιγράφετε το περιβάλλον εκπαίδευσης των αλγορίθμων. Επιπλέον στο Κεφάλαιο 6 παρουσιάζονται και σχολιάζονται τα αποτελέσματά της μάθησης. Και τέλος προτείνονται μελλοντικές επεκτάσεις.

Κεφάλαιο 2

Βασικές έννοιες της Ενισχυτικής Μάθησης

Τα βασικά χαρακτηριστικά της ενισχυτικής μάθησης είναι ο πράκτορας και το περιβάλλον. Το περιβάλλον είναι ο κόσμος στον οποίο ζει και αλληλεπιδρά ο πράκτορας. Σε κάθε βήμα αλληλεπίδρασης, ο πράκτορας βλέπει μια (ενδεχομένως μερική) παρατήρηση της κατάστασης του κόσμου και στη συνέχεια αποφασίζει για μια ενέργεια που πρέπει να κάνει. Το περιβάλλον αλλάζει όταν ο πράκτορας ενεργεί σε αυτό, αλλά μπορεί να αλλάξει και από μόνο του.

Ο πράκτορας μπορεί να αντιληφθεί επίσης ένα σήμα ανταμοιβής από το περιβάλλον, έναν αριθμό που του λέει πόσο καλή ή κακή είναι η τρέχουσα κατάσταση του κόσμου. Ο στόχος του πράκτορα είναι να μεγιστοποιήσει τη συνολική ανταμοιβή του, που ονομάζεται απόδοση. Οι μέθοδοι ενισχυτικής μάθησης είναι οι τρόποι με τους οποίους ο πράκτορας μαθαίνει συμπεριφορές για την επίτευξη του στόχου του. [21]



Σχήμα 2.1: *Αλληλεπίδραση Πράκτορα - Περιβάλλοντος [21]*

2.0.1 Καταστάσεις και παρατηρήσεις

Μια κατάσταση s είναι μια πλήρης περιγραφή του κόσμου. Δεν υπάρχει κάποια πληροφορία για τον κόσμο που να είναι κρυμμένη από την κατάσταση. Μια παρατήρηση ω είναι μια μερική περιγραφή μιας κατάστασης, η οποία μπορεί να παραλείπει πληροφορίες.

Στη βαθιά ενισχυτική μάθησή, σχεδόν πάντα αναπαριστούμε τις καταστάσεις και τις παρατηρήσεις με ένα διάνυσμα πραγματικών τιμών, έναν πίνακα ή έναν τανυστή ανώτερης τάξης.

Όταν ο πράκτορας είναι σε θέση να παρατηρήσει την πλήρη κατάσταση του περιβάλλοντος, λέμε ότι το περιβάλλον είναι πλήρως παρατηρήσιμο. Όταν ο πράκτορας μπορεί να δει

μόνο μια μερική παρατήρηση, λέμε ότι το περιβάλλον είναι μερικώς παρατηρήσιμο. [22]

2.0.2 Χώροι δράσης

Διαφορετικά περιβάλλοντα επιτρέπουν διαφορετικά είδη δράσεων. Το σύνολο όλων των έγκυρων ενεργειών σε ένα δεδομένο περιβάλλον ονομάζεται συχνά χώρος δράσεων. Ορισμένα περιβάλλοντα, όπως το Super Mario και το Go, έχουν διακριτούς χώρους δράσης, όπου μόνο ένας πεπερασμένος αριθμός κινήσεων είναι διαθέσιμος στον πράκτορα. Άλλα περιβάλλοντα, όπως όταν ο πράκτορας ελέγχει ένα ρομπότ σε έναν φυσικό κόσμο, έχουν συνεχείς χώρους δράσης. Στους συνεχείς χώρους, οι ενέργειες είναι διανύσματα πραγματικών τιμών.

Αυτή η διάκριση έχει κάποιες αρκετά σημαντικές συνέπειες για τις μεθόδους στη βαθιά ενισχυτική μάθησή. Ορισμένες οικογένειες αλγορίθμων μπορούν να εφαρμοστούν άμεσα μόνο στη μία περίπτωση ενώ θα πρέπει να επανασχεδιαστούν σημαντικά για την άλλη. [22]

2.1 Μαρκοβιανές διαδικασίες αποφάσεων

Μια στοχαστική διαδικασία ελέγχου διακριτού χρόνου είναι μαρκοβιανή (δηλ. έχει την ιδιότητα Markov) εάν

$$P(\omega_{t+1}|\omega_t, a_t) = P(\omega_{t+1}|\omega_t, a_t, ..., , \omega_0, a_0), \tag{2.1}$$

και

$$P(r_t|\omega_t, a_t) = P(r_t|\omega_t, a_t, ..., , \omega_0, a_0). \tag{2.2}$$

Θεωρώντας ένα πεπερασμένο σύνολο καταστάσεων, μια κατάσταση S_t ονομάζεται Μαρκοβιανή ή λέγεται ότι ικανοποιεί τη Μαρκοβιανή ιδιότητα εάν και μόνο εάν η επόμενη κατάσταση του περιβάλλοντος εξαρτάται μόνο από την τρέχουσα. Ένα περιβάλλον, για το οποίο όλες οι καταστάσεις του συνόλου ικανοποιούν την ιδιότητα, καλείται Μαρκοβιανό. [23]

Ένα πρόβλημα ενισχυτικής μάθησης μπορεί να περιγραφεί σαν μια Μαρκοβιανή Διαδικασία Αποφάσεων (Markov Decision Process - MDP). Μια MDP έχει 5 ορούς,(S, A, R, P, γ) όπου

S : είναι το σύνολο όλων των έγκυρων καταστάσεων,

A : είναι το σύνολο όλων των έγκυρων ενεργειών,

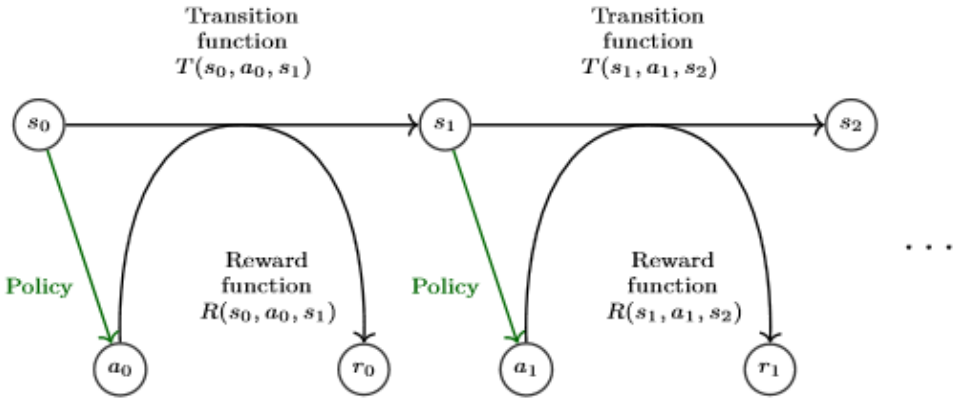
R : $S \times A \times S \to \mathcal{R}$ είναι η συνάρτηση ανταμοιβής, με $r_t = R(s_t, a_t, s_{t+1})$,

P : $S \times A \to \mathcal{P}(S)$ είναι η συνάρτηση πιθανότητας μετάβασης, με $P(s'|s, a)$ να είναι η πιθανότητα μετάβασης στην κατάσταση s' αν ξεκινήσουμε από την κατάσταση s και αναλάβουμε την ενέργεια a,

γ: είναι ένας παράγοντας έκπτωσης, όπου γε [0, 1)

Η ονομασία Διαδικασία Απόφασης Μαρκόφ αναφέρεται στο γεγονός, ότι το σύστημα υπακούει στην ιδιότητα Μαρκόφ: οι μεταβάσεις εξαρτώνται μόνο από την πιο πρόσφατη κατάσταση και δράση, και όχι από το προηγούμενο ιστορικό.

Το σύστημα είναι πλήρως παρατηρήσιμο σε ένα MDP, πράγμα που σημαίνει, ότι η παρατήρηση είναι ίδια με την κατάσταση του περιβάλλοντος: $\omega_t = s_t$. Σε κάθε χρονικό βήμα t, η πιθανότητα μετάβασης στην s_{t+1} δίνεται από τη συνάρτηση μετάβασης κατάστασης $T(s_t, a_t, s_{t+1})$ και η ανταμοιβή δίνεται από μια δεσμευμένη συνάρτηση ανταμοιβής $R(s_t, a_t, s_{t+1}) \in R$.



Σχήμα 2.2: *Απεικόνιση ενός MDP. Σε κάθε βήμα, ο πράκτορας εκτελεί μια ενέργεια που αλλάζει την κατάστασή του στο περιβάλλον και του αποφέρει μια ανταμοιβή. [21]*

2.1.1 Πολιτικές

Η πολιτική είναι το κριτήριο που χρησιμοποιεί ένας πράκτορας για να αποφασίσει ποιες ενέργειες πρέπει να κάνει. Μπορεί να είναι ντετερμινιστική, οπότε συνήθως συμβολίζεται με μ:

$$a_t = \mu(s_t), \tag{2.3}$$

ή μπορεί να είναι στοχαστική, οπότε συμβολίζεται συνήθως με π:

$$a_t \sim \pi(\cdot|s_t). \tag{2.4}$$

Στη βαθιά RL, ασχολούμαστε με παραμετροποιημένες πολιτικές: πολιτικές των οποίων οι έξοδοι είναι υπολογίσιμες συναρτήσεις που εξαρτώνται από ένα σύνολο παραμέτρων (π.χ. τα βάρη και οι προκαταλήψεις ενός νευρωνικού δικτύου), τις οποίες μπορούμε να προσαρμόσουμε για να αλλάξουμε τη συμπεριφορά μέσω κάποιου αλγορίθμου βελτιστοποίησης.

Συχνά συμβολίζουμε τις παραμέτρους μιας τέτοιας πολιτικής με θ ή ϕ και στη συνέχεια το γράφουμε ως δείκτη στο σύμβολο της πολιτικής για να τονίσουμε τη σύνδεση:

$$a_t = \mu_\theta(s_t) \tag{2.5}$$

$$a_t \sim \pi_\theta(\cdot|s_t). \tag{2.6}$$

Στοχαστικές πολιτικές

Τα δύο πιο συνηθισμένα είδη στοχαστικών πολιτικών στη βαθιά RL είναι οι κατηγορηματικές πολιτικές και οι διαγώνιες γκαουσιανές πολιτικές.

Οι κατηγορηματικές πολιτικές μπορούν να χρησιμοποιηθούν σε διακριτούς χώρους δράσης, ενώ οι διαγώνιες γκαουσιανές πολιτικές χρησιμοποιούνται σε συνεχείς χώρους δράσης.

Δύο βασικοί υπολογισμοί για τη χρήση και την εκπαίδευση στοχαστικών πολιτικών είναι:

- δειγματοληψία ενεργειών από την πολιτική,
- ο υπολογισμός των λογαριθμικών πιθανοτήτων συγκεκριμένων ενεργειών, $\log \pi_\theta(a|s)$.

[24]

Κατηγορηματικές πολιτικές

Μια κατηγορηματική πολιτική μοιάζει με έναν ταξινομητή πάνω σε διακριτές ενέργειες. Χτίζουμε το νευρωνικό δίκτυο για μια κατηγορική πολιτική με τον ίδιο τρόπο που θα κάναμε για έναν ταξινομητή: η είσοδος είναι η παρατήρηση, ακολουθείται από κάποιο αριθμό επιπέδων (πιθανώς συνελικτικών ή πυκνά συνδεδεμένων, ανάλογα με το είδος της εισόδου), και στη συνέχεια έχουμε ένα τελικό γραμμικό επίπεδο που μας δίνει λογάριθμους για κάθε ενέργεια, ακολουθούμενο από μια softmax για τη μετατροπή των λογαρίθμων σε πιθανότητες.

Συμβολίζουμε το τελευταίο επίπεδο πιθανοτήτων ως $P_\theta(s)$. Είναι ένα διάνυσμα με τόσες καταχωρήσεις όσες και οι ενέργειες, οπότε μπορούμε να θεωρήσουμε τις ενέργειες ως δείκτες για το διάνυσμα. Η λογαριθμική πιθανοφάνεια για μια ενέργεια a μπορεί στη συνέχεια να βρεθεί με το διάνυσμα:

$$log\pi_\theta(a|s) = \log [P_\theta(s)]_a \,. \tag{2.7}$$

Διαγώνιες Γκαουσιανές πολιτικές

Μια πολυμεταβλητή γκαουσιανή κατανομή (ή πολυμεταβλητή κανονική κατανομή) περιγράφεται από ένα μέσο διάνυσμα, μ, και έναν πίνακα συνδιακύμανσης, Σ. Μια διαγώνια κατανομή Gauss είναι μια ειδική περίπτωση, όπου ο πίνακας συνδιακύμανσης έχει καταχωρήσεις μόνο στη διαγώνιο. Ως αποτέλεσμα, μπορούμε να την αναπαραστήσουμε με ένα διάνυσμα.

Μια διαγώνια γκαουσιανή πολιτική έχει πάντα ένα νευρωνικό δίκτυο που απεικονίζει από τις παρατηρήσεις σε μέσες ενέργειες, $\mu_\theta(s)$. Υπάρχουν δύο διαφορετικοί τρόποι με τους οποίους αναπαρίσταται συνήθως ο πίνακας συνδιακύμανσης.

- Ο πρώτος τρόπος: Υπάρχει ένα ενιαίο διάνυσμα των λογαριθμικών τυπικών αποκλίσεων, $\log \sigma$, το οποίο δεν είναι συνάρτηση της κατάστασης: τα $\log \sigma$ είναι αυτόνομες παράμετροι. (Οι υλοποιήσεις των TRPO και PPO εφαρμόζουν αυτόν τον τρόπο).

- Ο δεύτερος τρόπος: Υπάρχει ένα νευρωνικό δίκτυο που αντιστοιχίζει από τις καταστάσεις σε λογαριθμικές τυπικές αποκλίσεις, $\log \sigma_\partial(s)$. Μπορεί προαιρετικά να μοιράζεται κάποια επίπεδα με το μέσο δίκτυο.

Και στις δύο περιπτώσεις εξάγουμε λογαριθμικές τυπικές αποκλίσεις αντί για τις τυπικές αποκλίσεις απευθείας.Αυτό οφείλεται στο γεγονός, ότι οι λογαριθμικές τυπικές αποκλίσεις είναι ελεύθερες να πάρουν οποιεσδήποτε τιμές στο $(-\infty, \infty)$, ενώ οι τυπικές αποκλίσεις πρέπει να είναι μη αρνητικές. Είναι ευκολότερο να εκπαιδεύσουμε παραμέτρους, αν δεν χρειάζεται να επιβάλλουμε τέτοιου είδους περιορισμούς. Οι τυπικές αποκλίσεις μπορούν να προκύψουν αμέσως από τις τυπικές αποκλίσεις λογαρίθμου με χρήση της εκθετικής, οπότε δεν χάνουμε τίποτα με την αναπαράστασή τους με αυτόν τον τρόπο.

Δειγματοληψία

Δεδομένης της μέσης δράσης $\mu_\partial(s)$ και της τυπικής απόκλισης $\sigma_\partial(s)$, και ενός διανύσματος z θορύβου από μια σφαιρική Γκαουσιανή($z \sim \mathcal{N}(0, I)$), ένα δείγμα δράσης μπορεί να υπολογιστεί ως εξής $a = \mu_\partial(s) + \sigma_\partial(s) \odot z$, όπου $\odot$ δηλώνει το κατά Hadamard γινόμενο δύο διανυσμάτων.

Λογαριθμική Πιθανότητα

Η λογαριθμική πιθανότητα μιας k -διάστατης δράσης a, για μια διαγώνια Γκαουσιανή με μέση τιμή $\mu = \mu_\partial(s)$ και τυπική απόκλιση $\sigma = \sigma_\partial(s)$, δίνεται από τη σχέση

$$log\pi_\partial(a|s) = -\frac{1}{2}\left(\sum_{i=1}^{k}\left(\frac{(a_i - \mu_i)^2}{\sigma_i^2} + 2\log\sigma_i\right) + k\log 2\pi\right). \tag{2.8}$$

2.1.2 Τροχιές

Μια τροχιά (trajectory) τ είναι μια ακολουθία καταστάσεων και ενεργειών στον κόσμο,

$$\tau = (s_0, a_0, s_1, a_1, ...). \tag{2.9}$$

Η πρώτη κατάσταση του κόσμου, s_0, λαμβάνεται τυχαία από την κατανομή της αρχικής κατάστασης, η οποία μερικές φορές συμβολίζεται με ρ_0:

$$s_0 \sim \rho_0(\cdot) \tag{2.10}$$

.

Οι μεταβάσεις κατάστασης, τι συμβαίνει στον κόσμο μεταξύ της κατάστασης τη στιγμή t, s_t,και της κατάστασης τη στιγμή $t+1, s_{t+1}$, διέπονται από τους φυσικούς νόμους του περιβάλλοντος και εξαρτώνται μόνο από την πιο πρόσφατη δράση, a_t. Μπορούν να είναι είτε ντετερμινιστικές,

$$s_{t+1} = f(s_t, a_t) \tag{2.11}$$

είτε στοχαστικές,

$$s_{t+1} \sim P(\cdot|s_t, a_t) \tag{2.12}$$

Οι ενέργειες προέρχονται από έναν πράκτορα σύμφωνα με την πολιτική του. Επίσης οι τροχιές ονομάζονται συχνά επεισόδια. [21]

2.1.3 Ανταμοιβή και επιστροφή

Η συνάρτηση ανταμοιβής ή αλλιώς επιστροφής(Return ή Reward function) R είναι εξαιρετικά σημαντική στην ενισχυτική μάθηση. Εξαρτάται από την τρέχουσα κατάσταση του κόσμου, την ενέργεια που μόλις έγινε και την επόμενη κατάσταση του κόσμου:

$$r_t = R(s_t, a_t, s_{t+1}) \tag{2.13}$$

Αν και συχνά απλοποιείται σε μια απλή εξάρτηση από την τρέχουσα κατάσταση,$r_t = R(s_t)$, ή σε ένα ζεύγος κατάστασης-δράσης $r_t = R(s_t, a_t)$. Ο στόχος του πράκτορα είναι να μεγιστοποιήσει κάποια έννοια της αθροιστικής ανταμοιβής σε μια τροχιά, αλλά αυτό στην πραγματικότητα μπορεί να εκφραστεί με διαφορετικούς τρόπους.

Ένα είδος ανταμοιβής είναι η μη προεξοφλημένη ανταμοιβή (undiscounted return) πεπερασμένου ορίζοντα, η οποία είναι απλώς το άθροισμα των ανταμοιβών που λαμβάνονται σε ένα σταθερό αριθμό βημάτων:

$$R(\tau) = \sum_{t=0}^{T} r_t. \tag{2.14}$$

Ένα άλλο είδος ανταμοιβής είναι η προεξοφλημένη ανταμοιβή (discounted return) απείρου ορίζοντα. Είναι βασικά ένα σταθμισμένο άθροισμα όλων των ανταμοιβών που πήρε ο πράκτορας κατά τη διάρκεια κάθε χρονικού βήματος στο τρέχον επεισόδιο, αλλά προεξοφλημένη ανάλογα με το πόσο μακριά στο μέλλον λαμβάνονται. Αυτή η διατύπωση της ανταμοιβής περιλαμβάνει έναν συντελεστή προεξόφλησης γ στο (0,1):

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t. \tag{2.15}$$

Βοηθάει να ξαναγράψουμε την προεξοφλημένη ανταμοιβή R ενός πράκτορα ως εξής

$$R(\tau) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k} = \sum_{\Delta t=0}^{\infty} e^{-\Delta t/\tau} r_{t+\Delta t} \tag{2.16}$$

όπου $\gamma = e^{-1/\tau}$ και $\mathrm{k} \to \Delta t$

Η τιμή τ δείχνει τον χρονικό ορίζοντα και συνδέεται άμεσα με τον συντελεστή προεξόφλησης γ, πχ για γ=1 έχουμε$\tau = \infty$. Οποιεσδήποτε ανταμοιβές βρίσκονται σε περισσότερα από τ χρονικά βήματα στο μέλλον καταστέλλονται εκθετικά. [20]

Από μαθηματική άποψη: ένα άπειρο άθροισμα ανταμοιβών μπορεί να μην συγκλίνει σε μια πεπερασμένη τιμή και είναι δύσκολο να αντιμετωπιστεί σε εξισώσεις, αλλά με έναν συντελεστή προεξόφλησης και υπό λογικές συνθήκες, το άπειρο άθροισμα συγκλίνει.

Διαισθητικά, ο συντελεστής προεξόφλησης ουσιαστικά καθορίζει πόσο πολύ οι πράκτορες ενισχυτικής μάθησης ενδιαφέρονται για τις ανταμοιβές στο μακρινό μέλλον σε σχέση με εκείνες στο άμεσο μέλλον. Εάν το γ είναι πολύ μικρο, ο πράκτορας θα είναι εντελώς μυωπικός και θα μαθαίνει μόνο για ενέργειες που παράγουν άμεση ανταμοιβή. Εάν το γ είναι πολύ μεγάλο, ο πράκτορας θα αξιολογεί κάθε ενέργειά του με βάση το άθροισμα όλων των μελλοντικών ανταμοιβών του. Δε θέλουμε το γ να είναι όσο το δυνατόν υψηλότερο, καθώς οι περισσότερες ενέργειες δεν έχουν μακροχρόνιες επιπτώσεις. Με μεγαλύτερο χρονικό ορίζοντα θα συνυπολογίζονται πολλές άσχετες πληροφορίες και, ως εκ τούτου, η συνάρτησή ανταμοιβής θα έχει τεράστια απόκλιση. Θα πρέπει γενικά να επιλέγεται ένας συντελεστής προεξόφλησης τέτοιος ώστε ο χρονικός ορίζοντας να περιέχει όλες τις σχετικές ανταμοιβές για μια συγκεκριμένη ενέργεια, αλλά όχι περισσότερες.

2.1.4 Συνάρτηση αξίας

Η συνάρτησή αξίας (baseline ή value function) προσπαθεί να δώσει μια εκτίμηση του προεξοφλημένου αθροίσματος ανταμοιβων που είδαμε από πάνω. Προσπαθεί να μαντέψει ποια θα είναι η τελική ανταμοιβή σε αυτό το επεισόδιο ξεκινώντας από την τρέχουσα κατάσταση. Αυτό είναι ένα πρόβλημα εποπτευόμενης μάθησης που λαμβάνει τις καταστάσεις ως είσοδο και ως έξοδο το νευρωνικό δίκτυο προσπαθεί να προβλέψει ποιο θα είναι το προεξοφλημενο άθροισμα ανταμοιβών από αυτήν την κατάσταση και μετά. Κατά τη διάρκεια της εκπαίδευσης το νευρωνικό δίκτυο που αντιπροσωπεύει την συνάρτησή άξιας θα ενημερώνεται συχνά χρησιμοποιώντας την εμπειρία που, ο πράκτορας συλλέγει από το περιβάλλον. Επειδή η εκτίμηση της άξιας είναι η έξοδος ενός νευρωνικού δικτύου θα αποκλίνει, καθώς το δίκτυό μας δεν πρόκειται να προβλέπει πάντα την ακριβή τιμή αυτών των καταστάσεων.

Οι συναρτήσεις αξίας χρησιμοποιούνται, σχεδόν σε κάθε αλγόριθμο ενισχυτικής μάθησης. Θα αναφέρουμε τέσσερις από αυτές:

- Η Συνάρτηση Αξίας Εντός Πολίτικης, $V^{\pi}(s)$, η οποία δίνει την αναμενόμενη ανταμοιβή αν ξεκινήσουμε στην κατάσταση s και ενεργούμε πάντα σύμφωνα με την πολιτική π:

$$V^{\pi}(s) = E_{\tau\sim\pi}[R(\tau)|s_0 = s] \tag{2.17}$$

- Η συνάρτηση Δράσης-Αξίας Εντός Πολίτικης, $Q^{\pi}(s,a)$, η οποία δίνει την αναμενόμενη ανταμοιβή αν ξεκινήσουμε στην κατάσταση s, λάβουμε μια αυθαίρετη δράση a (η οποία μπορεί να μην έχει προέλθει από την πολιτική), και στη συνέχεια ενεργούμε για πάντα σύμφωνα με την πολιτική π:

$$Q^{\pi}(s,a) = E_{\tau\sim\pi}[R(\tau)|s_0 = s, a_0 = a] \tag{2.18}$$

- Η Συνάρτηση Βέλτιστης Αξίας, $V^{*}(s)$, η οποία δίνει την αναμενόμενη ανταμοιβή αν ξεκινήσουμε στην κατάσταση s και ενεργούμε πάντα σύμφωνα με τη βέλτιστη πολιτική στο περιβάλλον:

$$V^*(s) = \max_{\pi} E_{\tau\sim\pi}[R(\tau)|s_0 = s] \tag{2.19}$$

- Η Συνάρτηση Βέλτιστης Δράσης-Αξίας, $Q^*(s,a)$, η οποία δίνει την αναμενόμενη ανταμοιβή αν ξεκινήσουμε στην κατάσταση s, λάβουμε μια αυθαίρετη δράση a, και στη συνέχεια ενεργούμε για πάντα σύμφωνα με τη βέλτιστη πολιτική στο περιβάλλον:

$$Q^*(s,a) = \max_{\pi} E_{\tau\sim\pi}[R(\tau)|s_0 = s, a_0 = a] \tag{2.20}$$

[20]

Από τους ορισμούς που δώσαμε προκύπτουν δύο βασικές συνδέσεις μεταξύ της συνάρτησης αξίας και της συνάρτησης δράσης-αξίας:

$$V^{\pi}(s) = E_{a\sim\pi}[Q^{\pi}(s,a)], \tag{2.21}$$

$$V^*(s) = \max_{a} Q^*(s,a). \tag{2.22}$$

Υπάρχει μια σημαντική σύνδεση μεταξύ της βέλτιστης συνάρτησης $Q^*(s,a)$ και της δράσης που επιλέγεται από τη βέλτιστη πολιτική. Εξ ορισμού, η $Q^*(s,a)$ δίνει την αναμενόμενη απόδοση για την εκκίνηση στην κατάσταση s, την ανάληψη (αυθαίρετης) δράσης a, ώστε στη συνέχεια να ενεργούμε σύμφωνα με τη βέλτιστη πολιτική για πάντα.

Η βέλτιστη πολιτική στην s θα επιλέξει όποια ενέργεια μεγιστοποιεί την αναμενόμενη απόδοση από την εκκίνηση στην s. Ως αποτέλεσμα, αν έχουμε Q^*, μπορούμε να λάβουμε άμεσα τη βέλτιστη ενέργεια, $a^*(s)$, μέσω

$$a^*(s) = \arg\max_{a} Q^*(s,a). \tag{2.23}$$

Μπορεί να υπάρχουν πολλαπλές ενέργειες που μεγιστοποιούν το $Q^*(s,a)$, οπότε όλες είναι βέλτιστες και η βέλτιστη πολιτική μπορεί να επιλέγει τυχαία οποιαδήποτε από αυτές. Υπάρχει, όμως, πάντα μια βέλτιστη πολιτική που επιλέγει ντετερμινιστικά μια ενέργεια.
[20]

Εξισώσεις Bellman

Και οι τέσσερις συναρτήσεις αξίας υπακούουν σε ειδικές εξισώσεις αυτοσυνέπειας που ονομάζονται εξισώσεις Bellman. Η βασική ιδέα πίσω από τις εξισώσεις Bellman είναι η εξής:

Η αξία του σημείου εκκίνησης είναι η ανταμοιβή που περιμέναμε να πάρουμε από την παρουσία μας εκεί, συν την αξία του σημείου όπου θα προσγειωθούμε στη συνέχεια. Έτσι η εξίσωση Bellman αναλύει τη συνάρτηση αξίας σε δύο μέρη, την άμεση ανταμοιβή συν τις προεξοφλημένες μελλοντικές αξίες. Η εξίσωση αυτή απλοποιεί τον υπολογισμό της συνάρτησης αξίας, αντί να αθροίζουμε σε πολλαπλά χρονικά βήματα, μπορούμε να βρούμε τη βέλτιστη λύση ενός πολύπλοκου προβλήματος αναλύοντάς το σε απλούστερα, αναδρομικά υποπροβλήματα και βρίσκοντας τις βέλτιστες λύσεις τους.

Οι εξισώσεις Bellman για τις συναρτήσεις αξίας εντός πολιτικής είναι οι εξής:

$$V^{\pi}(s) = E_{a\sim\pi, s'\sim P} r(s,a) + \gamma V^{\pi}(s'), \tag{2.24}$$

$$Q^{\pi}(s,a) = E_{s'\sim P} r(s,a) + \gamma E_{a'\sim\pi} Q^{\pi}(s',a'), \tag{2.25}$$

όπου $s' \sim P$ είναι συντομογραφία για $s' \sim P(\cdot|s,a)$, υποδεικνύοντας ότι η επόμενη κατάσταση s' επιλέγεται από τους κανόνες μετάβασης του περιβάλλοντος-$a \sim \pi$ είναι συντομογραφία για $a \sim \pi(\cdot|s)$ και $a' \sim \pi$ είναι συντομογραφία για $a' \sim \pi(\cdot|s')$.

Οι εξισώσεις Bellman για τις βέλτιστες συναρτήσεις αξίας είναι

$$V^{*}(s) = \max_{a} E_{s'\sim P} r(s,a) + \gamma V^{*}(s'), \tag{2.26}$$

$$Q^{*}(s,a) = E_{s'\sim P} r(s,a) + \gamma \max_{a'} Q^{*}(s',a'). \tag{2.27}$$

Η κρίσιμη διαφορά μεταξύ των εξισώσεων Bellman για τις συναρτήσεις αξίας εντός πολιτικής και τις βέλτιστες συναρτήσεις αξίας, είναι η απουσία ή η παρουσία της max επί των ενεργειών. Η συμπερίληψή της αντικατοπτρίζει το γεγονός, ότι κάθε φορά που ο πράκτορας επιλέγει την ενέργειά του, προκειμένου να ενεργήσει βέλτιστα, πρέπει να επιλέξει όποια ενέργεια οδηγεί στην υψηλότερη τιμή.

2.1.5 Συνάρτηση πλεονεκτήματος

Μερικές φορές στην ενισχυτική μάθησή, δεν χρειάζεται να περιγράψουμε πόσο καλή είναι μια ενέργεια με απόλυτη έννοια, αλλά μόνο πόσο καλύτερη είναι από άλλες κατά μέσο όρο. Δηλαδή, θέλουμε να γνωρίζουμε το σχετικό πλεονέκτημα αυτής της ενέργειας. Προσδιορίζουμε με ακρίβεια αυτή την έννοια με τη συνάρτηση πλεονεκτήματος (advantage function.). Η συνάρτηση πλεονεκτήματος προσπαθεί να εκτιμήσει ποια είναι η άξια της επιλεγμένης δράσης στην τρέχουσα κατάσταση.

Η συνάρτηση πλεονεκτήματος $A^{\pi}(s,a)$ που αντιστοιχεί σε μια πολιτική π περιγράφει πόσο καλύτερο είναι να ακολουθήσετε μια συγκεκριμένη ενέργεια a στην κατάσταση s, σε σχέση με την τυχαία επιλογή μιας ενέργειας σύμφωνα με την $\pi(\cdot|s)$, υποθέτοντας ότι ενεργείτε σύμφωνα με την π για πάντα μετά. Η συνάρτηση πλεονεκτήματος ορίζετε ως:

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s). \tag{2.28}$$

2.2 Δυναμικός προγραμματισμός

Ένα μοντέλο είναι η αναπαράσταση του περιβάλλοντος από έναν πράκτορα, συμπεριλαμβανομένου του συστήματος μετάβασης καταστάσεων και του συστήματος ανταμοιβής. Όταν ένα μοντέλο είναι διαθέσιμο για το πρόβλημα μας, δηλαδή γνωρίζουμε τις πιθανότητες μετάβασης P_s μεταξύ των διάφορων καταστάσεων, τότε μπορούμε να εφαρμόσουμε μεθόδους Δυναμικού Προγραμματισμού, ΔΠ (Dynamic Programming,DP) για την επίλυσή των συναρτήσεων.

Ο δυναμικός προγραμματισμός είναι μια μέθοδος επίλυσης σύνθετων προβλημάτων με τη διάσπασή τους σε υποπροβλήματα. Οι λύσεις των υποπροβλημάτων συνδυάζονται για την

επίλυση του συνολικού προβλήματος. Η ιδέα του δυναμικού προγραμματισμού είναι ότι δεν χρειάζεται να λύσουμε ένα πρόβλημα που έχουμε ήδη λύσει. [21]

Οι δύο απαιτούμενες ιδιότητες του δυναμικού προγραμματισμού είναι οι εξής:

- Βέλτιστη υποδομή: η βέλτιστη λύση του υποπροβλήματος μπορεί να χρησιμοποιηθεί για την επίλυση του συνολικού προβλήματος.
- Επικαλυπτόμενα υποπροβλήματα: τα υποπροβλήματα επαναλαμβάνονται πολλές φορές. Οι λύσεις των υποπροβλημάτων μπορούν να αποθηκευτούν και να επαναχρησιμοποιηθούν.

Οι Διαδικασίες Αποφάσεων Μαρκόφ ικανοποιούν και τις δύο αυτές ιδιότητες:

- Η Εξίσωση Bellman μας δίνει ότι η αξία μιας κατάστασης είναι ίση με την άμεση ανταμοιβή που παίρνει ο πράκτορας μας φεύγοντας από την κατάσταση συν την αξία της επόμενης κατάστασης. Έτσι, η εξίσωση αναλύει τη διαδικασία εύρεσης της συνάρτησης αξίας μιας κατάστασης, χωρίζοντάς την σε υποπροβλήματα.
- Οι συναρτήσεις αξίας έχουν ήδη αποθηκεύσει πόσο καλή είναι μια συγκεκριμένη κατάσταση, ώστε να μη χρειάζεται να υπολογίζουμε ξανά και ξανά την αξία αυτής της κατάστασης. Για παράδειγμα, ας υποθέσουμε ότι υπάρχουν δύο καταστάσεις(s_1 και s_2) και βρισκόμαστε στην κατάσταση s_1 και έχουμε ήδη υπολογίσει την τιμή της κατάστασης s_2, οπότε κατά τον υπολογισμό της τιμής της κατάστασης s_1 δε θα υπολογίσουμε εκ νέου την τιμή της κατάστασης s_2. (Η τιμή τηςs_1 βασίζεται στην κατάσταση s_2)

[20]

Ο δυναμικός προγραμματισμός μπορεί να χρησιμοποιηθεί για την επίλυση προβλημάτων ενισχυτικής μάθησης όταν κάποιος μας δίνει τη δομή του MDP (δηλαδή όταν γνωρίζουμε τη δομή της μετάβασης, τη δομή της ανταμοιβής κ.λπ.). Επομένως ο δυναμικός προγραμματισμός χρησιμοποιείται για τον σχεδιασμό (planning) σε ένα MDP είτε για την επίλυση:

- Πρόβλημα πρόβλεψης (Prediction problem) (αξιολόγηση πολιτικής):

  Δίνεται ένα MDP <S, A, P, R, γ> και μια πολιτική π. Βρίσκουμε τη συνάρτηση αξίας V_π (η οποία μας λέει πόση ανταμοιβή θα πάρουμε σε κάθε κατάσταση). δηλ. ο στόχος είναι να βρούμε πόσο καλή είναι μια πολιτική π.

- Πρόβλημα ελέγχου (Control problem) (Βρίσκουμε την καλύτερή δράση στο MDP):

  Δίνεται ένα MDP <S, A, P, R, γ>. Βρίσκουμε τη βέλτιστη συνάρτηση αξίας V_π και τη βέλτιστη πολιτική π*. δηλ. ο στόχος είναι να βρούμε την πολιτική που δίνει τη μεγαλύτερη ανταμοιβή που μπορούμε να λάβουμε με την καλύτερη ενέργεια που μπορούμε να επιλέξουμε. [25]

2.2.1 Αξιολόγηση πολιτικής

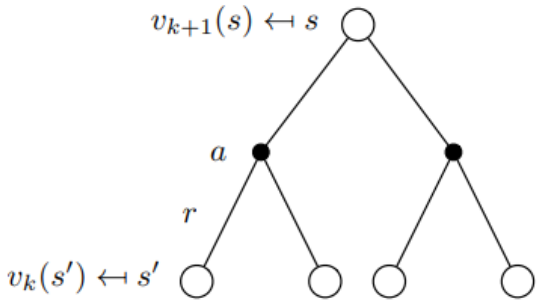
Αξιολόγηση πολιτικής σημαίνει πόση ανταμοιβή θα πάρει ο πράκτοράς μας ακολουθώντας μια συγκεκριμένη πολιτική π. Αξιολογούμε μια δεδομένη πολιτική π χρησιμοποιώντας

την εξίσωση προσδοκιών Bellman. Η εξίσωση προσδοκίας Bellman λέει πόση ανταμοιβή θα πάρει ο πράκτοράς μας ακολουθώντας μια πολιτική π και ορίζεται ως εξής:

$$V_{k+1} = \sum_{a \in A} \pi(a|s)(R_s^a + \gamma + \sum_{s' \in S} P_{ss'}^a V_k(s')) \tag{2.29}$$

Για να αξιολογήσουμε μια πολιτική θα χρησιμοποιήσουμε αυτή την εξίσωση με επαναληπτικό τρόπο. Αυτό σημαίνει ότι θα υπολογίσουμε την τιμή της επόμενης κατάστασης με την υποστήριξη της τιμής της τρέχουσας κατάστασης από την προηγούμενη επανάληψη. Ξεκινάμε με τη συνάρτηση αρχικών τιμών V_1 (μια τιμή όλων των καταστάσεων στο MDP). Π.χ. αρχίζουμε με την τιμή 0. Επομένως, δεν υπάρχει ανταμοιβή. Στη συνέχεια χρησιμοποιούμε την εξίσωση προσδοκίας Bellman για τον υπολογισμό της V_2 και επαναλαμβάνουμε πολλές φορές, τελικά θα συγκλίνει στην V_π.

$$V_1 \leftarrow V_2 \leftarrow V_3 \leftarrow ...V_\pi \leftarrow \tag{2.30}$$



Σχήμα 2.3: *Διάγραμμα για τον υπολογισμό της συνάρτησης αξίας της ρίζας.*

Για παράδειγμά για το δεντρο 2.3 γνωρίζουμε ότι η τιμή του ριζικού κόμβου s δίνεται από την ανταμοιβή που πήραμε με την ενέργεια a συν την τιμή της κατάστασης s'. Τώρα, αυτό που κάνουμε είναι ότι χρησιμοποιούμε την εξίσωση Bellman Expectation με επαναληπτικό τρόπο για δυναμικό προγραμματισμό. Δηλαδή, θα χρησιμοποιήσουμε την τιμή των φύλλων (κατάσταση s' εδώ) που είναι από την προηγούμενη επανάληψη, όταν υπολογίζαμε την τιμή της κατάστασης s, για να υπολογίσουμε την τιμή της κατάστασης s. Αυτό συμβαίνει σε κάθε επανάληψη. Τώρα, για τη δεύτερη επανάληψη, θα χρησιμοποιήσουμε την τιμή της κατάστασης που υπολογίσαμε στην προηγούμενη επανάληψη που είναι v_{k+1} για να υπολογίσουμε την τιμή της επόμενης κατάστασης, δηλαδή v_{k+2}.

Αποδεικνύεται ότι αν αρχίσουμε να συμπεριφερόμαστε άπληστα ως προς τις τιμές των καταστάσεων θα καταλήξουμε τελικά σε μια βέλτιστη πολιτική. Το να συμπεριφερόμαστε άπληστα σημαίνει ότι από όλες τις καταστάσεις στις οποίες μπορεί να πάει ο πράκτοράς μας, επιλέγει αυτή με τη μεγαλύτερη αξία από τις υπόλοιπες.

2.2.2 Προσέγγισή πολιτικής

Αυτός ο αλγόριθμος χωρίζει τη διαδικασία που περιγράφηκε προηγουμένως σε δύο μέρη.

Πρώτον, θα αξιολογήσουμε μια πολιτική π, χρησιμοποιώντας την εξίσωση προσδοκίας Bellman, όπως περιγράφηκε στην προηγούμενη ενότητα, και στη συνέχεια θα ενεργήσουμε άπληστα σε αυτή την αξιολογημένη συνάρτηση αξίας για να βρούμε μια βέλτιστη πολιτική. Θα επαναλάβουμε αυτή τη διαδικασία μέχρι να βρούμε την πραγματική συνάρτηση αξίας.

Η ιδέα της προσέγγισή πολιτικής αποτελείται από δύο βήματα:

Αξιολόγηση πολιτικής (όπως περιγράφηκε προηγουμένως)

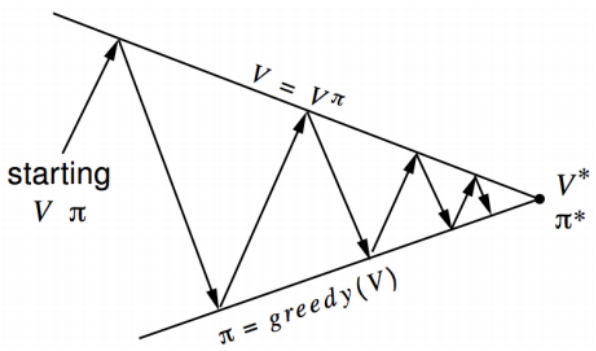
$$V_\pi(s) = E[R_{t+1} + \gamma R_{t+2} + ...|S_t = s] \tag{2.31}$$

Ενεργώντας άπληστα στην αξιολογημένη Συνάρτηση Αξίας αποδίδει μια πολιτική καλύτερη από την προηγούμενη

$$\pi' = greedy(V_\pi) \tag{2.32}$$

Επαναλαμβάνουμε αυτές τις δύο διαδικασίες έως ότου συγκλίνουν στη βέλτιστη συνάρτηση αξίας και στη βέλτιστη πολιτική.

Μετά την αξιολόγηση μιας πολιτικής (υπολογισμός της συνάρτησης αξίας χρησιμοποιώντας την εξίσωση προσδοκίας Bellman με επαναληπτικό τρόπο), ενεργούμε άπληστα σε αυτή τη συνάρτηση αξίας και επειδή ενεργούμε άπληστα αυτό καθιστά την πολιτική μας ντετερμινιστική. [8]



Σχήμα 2.4: *Policy Iteration.*

Στο Σχήμα 2.4 τα πάνω βέλη υποδεικνύουν την αξιολόγηση της πολιτικής και το κάτω βέλος υποδεικνύει ότι ενεργούμε άπληστα στη συνάρτηση αξίας. Το σχήμα μας εξηγεί ότι ξεκινάμε με τυχαίες τιμές σε κάθε κατάσταση (με μηδενικά) και με μια τυχαία πολιτική και στη συνέχεια αξιολογούμε αυτή την πολιτική χρησιμοποιώντας την εξίσωση προσδοκίας Bellman (Bellman Expectation) όπως περιγράφηκε προηγουμένως και στη συνέχεια ενεργούμε άπληστα στη συνάρτηση αξίας η οποία μας δίνει μια νέα πολιτική. Αξιολογούμε και πάλι αυτή τη νέα πολιτική, ενεργούμε και πάλι άπληστα στη νέα συνάρτηση αξίας. Συνεχίζουμε αυτή τη διαδικασία μέχρι να βρούμε τη βέλτιστη συνάρτηση αξίας και τη βέλτιστη πολιτική για τη διαδικασία απόφασης Markov.

Το να ενεργούμε άπληστα σημαίνει ότι επιλέγουμε μια ενέργεια, ας πούμε a, που θα μας αποφέρει τη μέγιστη αξία σε μια συγκεκριμένη κατάσταση s. Μπορούμε να το ορίσουμε ως εξής:

$$\pi'(s) = arg\max_{a \in A} q_\pi(s,a) \tag{2.33}$$

Το $q_\pi(s,a)$ μας λέει πόσο καλό είναι να αναλάβουμε την ενέργεια a στην κατάσταση s ή πόση αξία θα πάρουμε αναλαμβάνοντας την ενέργεια a στην κατάσταση s. Το να κάνουμε arg-max πάνω στο $q_\pi(s,a)$ σημαίνει ότι διαλέγουμε ενέργειες π.χ. a στην κατάσταση s που μεγιστοποιούν την q-αξία $q_\pi(s,a)$ ή μας αποδίδουν τη μέγιστη q-αξία $q_\pi(s,a)$. Παίρνοντας το πρώτο βήμα "άπληστα" π'(s) και στη συνέχεια ακολουθώντας τη πολιτική π είναι καλύτερο ή ίσο με το να ακολουθούσαμε απλώς την πολιτική π σε όλη τη διάρκεια της διαδικασίας.Αυτό μπορεί να εκφραστεί ως εξής:

$$p_\pi(s,\pi'(s)) = \max_{a \in A} q_\pi(s,a) \geq q_\pi(s,\pi(s)) = \upsilon_\pi(s) \tag{2.34}$$

[25]

Εδώ, η αριστερή πλευρά κάνει πρώτα το άπληστο βήμα μας [π'(s)] και στη συνέχεια ακολουθεί την πολιτική μας π. Η οποία ισούται με τη λήψη της καλύτερης δυνατής ενέργειας (η οποία έχει τη μέγιστη τιμή q). Η δεξιά πλευρά απλώς ακολουθεί μια πολιτική π σε όλη τη διάρκεια, η οποία ισούται με το να λέμε τη συνάρτηση αξίας μιας κατάστασης. Είναι καλύτερο ή ίσο από το να ακολουθήσουμε την πολιτική π από την αρχή καθώς βλέπουμε, ότι όταν επιλέγουμε το άπληστο βήμα π'(s) στο επόμενο βήμα επιλέγουμε την καλύτερη ενέργεια που μπορεί να ληφθεί από την κατάσταση s. Στη δεξιά πλευρά, δεν παίρνουμε κανένα άπληστο βήμα ή ντετερμινιστικό βήμα, οπότε υπάρχει πιθανότητα ο πράκτορας μας να επιλέξει την καλύτερη ενέργεια ή να επιλέξει κάποια ενέργεια που αποδίδει λιγότερη αξία από την άπληστη.Η ίδια ιδέα μπορεί να εφαρμοστεί σε κάθε βήμα:

$$\begin{aligned}
V_\pi(s) \leq q_\pi(s,\pi(s')) = \\
E'_\pi[R_{t+1} + \gamma_V \pi(S_{t+1})|S_t = s] \leq \\
E'_\pi[R_{t+1} + \gamma q_\pi(St+1, \pi'(S_{t+1})|S_t = s] \leq \\
E'_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 q_\pi(S_t+2, \pi'(S_{t+2})|S_t = s] \leq \\
E'_\pi[R_{t+1} + \gamma R_{t+2} + ...|S_t = s] = V_{\pi'}(s)
\end{aligned} \tag{2.35}$$

Έτσι, η πολιτική γίνεται καλύτερη σε κάθε βήμα και σταματά είναι όταν το π γίνει η βέλτιστη πολιτική μας. Έτσι, αν η συνάρτηση αξίας $V_\pi(s)$ είναι ίση με την άπληστη συνάρτηση αξίας $q_\pi(s,\pi'(s))$ τότε η Βέλτιστη Εξίσωση Bellman ικανοποιείται:

$$V_\pi(s) = \max_{a \in A} q_\pi(s,a) \tag{2.36}$$

2.2.3 Προσέγγιση Αξίας

Ένα μειονέκτημα της προσέγγιση πολιτικής είναι ότι αξιολογεί την πολιτική σε κάθε βήμα. Αντ' αυτού μπορούμε να εφαρμόσουμε προσέγγισή Άξιας (Value iteration): Αρχικοποίηση τιμών αυθαίρετα, π.χ. V0(s) = 0 για κάθε s Οι διαδοχικές προσεγγίσεις/βελτιώσεις

βασίζονται στην

$$V_{k+1}(s) = \max_a \sum_{s',r} p(s', r|s, a)(r + \gamma V_k(s')) \tag{2.37}$$

Σταματάμε όταν η συνάρτηση αξίας σταματά να μεταβάλλεται

$$\max_s |V_{k+1}(s) - V_k(s)| < \theta \tag{2.38}$$

Στην προσέγγισή πολιτικής, είδαμε ότι πρώτα αξιολογούμε μια πολιτική, δηλαδή βρίσκουμε τη συνάρτηση αξίας και στη συνέχεια ενεργούμε άπληστα σε αυτήν για να κατασκευάσουμε μια νέα πολιτική. Όμως, στην προσέγγισή αξίας δεν υπολογίζουμε την πολιτική για κάθε ενδιάμεση συνάρτηση αξίας. Μόλις βρούμε τη βέλτιστη συνάρτηση αξίας, τότε υπολογίζουμε τη βέλτιστη πολιτική από την τελική βέλτιστη συνάρτηση αξίας. Η προσέγγισή αξίας είναι η ειδική περίπτωση της προσέγγισή πολιτικής.

Για την προσέγγισή Αξίας χρησιμοποιείται η Βέλτιστη Εξίσωση Bellman. Ξεκινάμε με κάποια τυχαία Συνάρτηση Αξίας (π.χ. μηδέν) και στη συνέχεια επαναλαμβάνουμε αυτή τη διαδικασία χρησιμοποιώντας τη Βέλτιστη Εξίσωση Bellman. Δηλαδή, υπολογίζουμε τη νέα συνάρτηση αξίας μιας κατάστασης εισάγοντας τις τιμές της προηγούμενης επανάληψης στη Βέλτιστη Εξίσωση Bellman. Επαναλαμβάνουμε αυτή τη διαδικασία μέχρι να συγκλίνει στη βέλτιστη συνάρτηση αξίας και στη συνέχεια υπολογίζουμε τη βέλτιστη πολιτική από αυτή τη βέλτιστη συνάρτηση αξίας. [8]

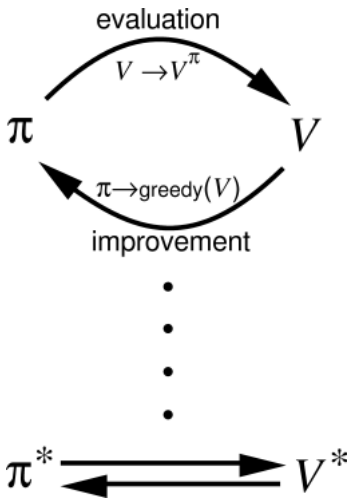
2.2.4 Ασύγχρονος δυναμικός προγραμματισμός

Το μειονέκτημα των μεθόδων DP είναι ότι περιλαμβάνουν πράξεις σε ολόκληρο το σύνολο καταστάσεων του MDP. Εάν ο χώρος καταστάσεων είναι πολύ μεγάλος, αυτό καθίσταται υπολογιστικά απαγορευτικό. Στον ασύγχρονο δυναμικό προγραμματισμό (Asynchronous dynamic programming) οι αλγόριθμοι ανανεώνουν τις άξιες των καταστάσεων με οποιαδήποτε σειρά , χρησιμοποιώντας οποιεσδήποτε τιμές άλλων καταστάσεων τυχαίνει να είναι διαθέσιμες. Κατά συνέπειά τιμές ορισμένων καταστάσεων μπορεί να χρησιμοποιηθούν αρκετές φορές προτού άλλες χρησιμοποιηθούν για πρώτη φορά. Εκμεταλλευόμαστε αυτή την ευελιξία επιλέγοντας τις καταστάσεις τις οποίες ενημερώνουμε ώστε να βελτιώσουμε το ρυθμό προόδου του αλγορίθμου. Μπορούμε να προσπαθήσουμε να διατάξουμε τις ενημερώσεις ώστε οι πληροφορίες να διαδίδονται από κατάσταση σε κατάσταση με αποτελεσματικό τρόπο. Ορισμένες καταστάσεις μπορεί να μην χρειάζονται ενημέρωση των τιμών τους τόσο συχνά όσο άλλες. Ακόμη μπορούμε και να προσπαθήσουμε να παραλείψουμε την ενημέρωση ορισμένων καταστάσεων εντελώς, εάν δεν είναι σχετικές με τη βέλτιστη συμπεριφορά

Γενικευμένη προσέγγισή πολιτικής (Generalised policy iteration, GPI)

Χρησιμοποιούμε τον όρο γενικευμένη προσέγγισή πολιτικής (GPI) για να αναφερθούμε στη γενική ιδέα να αφήνουμε τις διαδικασίες αξιολόγησης και βελτίωσης πολιτικής να αλληλεπιδρούν, ανεξάρτητα τις λεπτομέρειες των δύο διαδικασιών. Σχεδόν όλες οι μέθοδοι ενισχυτικής μάθησης περιγράφονται καλά ως GPI. Δηλαδή, όλες έχουν αναγνωρίσιμες πολιτικές και συναρτήσεις αξίας, με την πολιτική να βελτιώνεται πάντα σε σχέση με τη συνάρτηση

αξίας και τη συνάρτηση αξίας να οδηγείται πάντα προς τη συνάρτηση αξίας της πολιτικής. Αυτό απεικονίζεται στο Σχήμα 2.5



Σχήμα 2.5: *Οι συναρτήσεις αξίας και πολιτικής αλληλεπιδρούν μέχρι να είναι βέλτιστες και συνεπώς συνεπείς μεταξύ τους.*

Είναι εύκολο να διαπιστώσει κανείς ότι εάν τόσο η διαδικασία αξιολόγησης όσο και η διαδικασία βελτίωσης σταθεροποιούνται, δηλαδή δεν παράγουν πλέον αλλαγές, τότε η συνάρτηση αξίας και η πολιτική πρέπει να είναι βέλτιστες. Η συνάρτηση αξίας σταθεροποιείται μόνο όταν είναι συνεπής με την τρέχουσα πολιτική και η πολιτική σταθεροποιείται μόνο όταν είναι άπληστη σε σχέση με την τρέχουσα συνάρτηση αξίας. Αυτό σημαίνει ότι ισχύει η εξίσωση Bellman και επομένως ότι η πολιτική και η συνάρτηση αξίας είναι βέλτιστες.

Οι διαδικασίες αξιολόγησης και βελτίωσης της πολιτικής αλληλεπιδρούν ανεξαρτήτως της αναλυτικότητας των δύο διαδικασιών. Οι διαδικασίες αξιολόγησης και βελτίωσης στην GPI μπορεί να θεωρηθεί ότι:

- Ανταγωνίζονται υπό την έννοια ότι κινούνται προς αντίθετες κατευθύνσεις. Το να γίνει η πολιτική άπληστη σε σχέση με τη συνάρτηση αξίας συνήθως καθιστά τη συνάρτηση αξίας λανθασμένη για την αναθεωρημένη πολιτική, και το να γίνει η συνάρτηση αξίας συνεπής με την πολιτική τυπικά προκαλεί ότι η πολιτική αυτή δεν είναι πλέον άπληστη.

- Μακροπρόθεσμα, ωστόσο συνεργάζεται, οι δύο διαδικασίες αλληλεπιδρούν για να βρουν μια ενιαία κοινή λύση

Αποτελεσματικότητα του δυναμικού προγραμματισμού

Αν n και k συμβολίζουν τον αριθμό των καταστάσεων και των ενεργειών, ο συνολικός αριθμός των (ντετερμινιστικών) πολιτικών είναι k^n. Μια μέθοδος DP απαιτεί λιγότερες υπολογιστικές πράξεις από μια πολυωνυμική συνάρτηση των n και k. Ωστόσο ο DP έχει περιορισμένη εφαρμογή λόγω του προβλήματος της διαστατικότητας, του γεγονότος ότι ο αριθμός των καταστάσεων συχνά αυξάνεται εκθετικά με τον αριθμό των μεταβλητών κατάστασης.Βέβαιος αυτό είναι η εγγενής δυσκολία του προβλήματος, όχι του DP ως μεθόδου επίλυσης.

2.3 Ενισχυτική μάθηση χωρίς μοντέλα

Το πλήρες μοντέλο του περιβάλλοντος δεν είναι πάντα διαθέσιμο. Στην περίπτωση αυτή, ο πράκτορας μαθαίνει από την εμπειρία. Η εμπειρία μπορεί να αποκτηθεί από την αλληλεπίδραση με προσημειωμένο ή πραγματικό περιβάλλον. Ακόμη και αν ο πράκτορας έχει μια προσομοίωση περιβάλλοντος, μπορεί και πάλι να βρει τη βέλτιστη συμπεριφορά. [26]

2.3.1 Μέθοδοι Monte Carlo

Είναι πολλές οι περιπτώσεις, ιδιαίτερα στον πραγματικό κόσμο, όπου δεν έχουμε διαθέσιμο ένα πλήρες μοντέλο του προβλήματος, είτε το πλήθος των καταστάσεων είναι τόσο μεγάλο που καθιστά αδύνατη την εφαρμογή μεθόδων δυναμικού προγραμματισμού. Ένας τρόπος επίλυσής για αυτές τις περιπτώσεις είναι η χρήση της μεθόδου Monte Carlo. Η μέθοδος Monte Carlo αλληλεπιδρά με το περιβάλλον και χρησιμοποιεί τον εμπειρικό μέσο για την εκτίμηση των value functions. Για να καθορίσουμε τις αποδόσεις υποθέτουμε ότι η εμπειρία χωρίζεται σε επεισόδια, και ότι όλα τα επεισόδια τελικά τερματίζονται ανεξάρτητα από τις ενέργειες που επιλέγονται. Μόνο κατά την ολοκλήρωση ενός επεισοδίου αλλάζουν οι εκτιμήσεις των τιμών και οι πολιτικές. Συνεπώς, οι μέθοδοι Monte Carlo ενημερώνονται ανά επεισόδιο, αλλά όχι βήμα προς βήμα (online).

Οι μέθοδοι Monte Carlo δειγματοληπτούν και υπολογίζουν τον μέσο όρο των αποδόσεων για κάθε ζεύγος κατάστασης-δράσης. Η απόδοση μετά την ανάληψη μιας δράσης σε μια κατάσταση εξαρτάται από τις ενέργειες που αναλαμβάνονται σε μεταγενέστερες καταστάσεις του ίδιου επεισοδίου. Επειδή όλες οι επιλογές δράσης υποβάλλονται σε μάθηση, το πρόβλημα καθίσταται μη σταθερό σε σχέση με την προηγούμενη κατάσταση. Για να χειριστούμε τη αστάθεια, προσαρμόζουμε την ιδέα της προσέγγισή γενικής πολιτικής (GPI) που αναπτύχθηκε για το ΔΠ. Ενώ εκεί υπολογίσαμε συναρτήσεις αξίας από τη γνώση του MDP, εδώ μαθαίνουμε συναρτήσεις αξίας από δειγματικές αποδόσεις με το MDP. Η συνάρτηση αξίας και οι αντίστοιχες πολιτικές εξακολουθούν να αλληλεπιδρούν για να επιτύχουν τη βέλτιστη κατάσταση, ουσιαστικά με τον ίδιο τρόπο με την (GPI). Όπως και στο κεφάλαιο της ΔΠ, πρώτα εξετάζουμε το πρόβλημα πρόβλεψης (το υπολογισμό των v και q για μια σταθερή αυθαίρετη πολιτική) και στη συνέχεια τη βελτίωση της πολιτικής, και, τέλος, το πρόβλημα ελέγχου και η επίλυσή του από την GPI. Κάθε μία από αυτές τις ιδέες που λαμβάνονται από το DP επεκτείνεται στην περίπτωση Monte Carlo στην οποία είναι διαθέσιμη μόνο η δειγματοληπτική εμπειρία [20]

Η πρόβλεψη Monte Carlo εκτιμά τη συνάρτηση αξίας για μια δεδομένη πολιτική.

Αλγοριθμος 2.1: *Monte Carlo prediction*

Initialise V arbitrarily
Returns(s) ← empty list $\forall s \in S$
repeat
    Generate an episode using π
    for s in the episode **do**
        Returns(s) ← append return following s
    end for
    V (s) = average(Returns(s))
until convergence

Κάθε μέσος όρος είναι μια αμερόληπτη εκτίμηση, και η τυπική απόκλιση του σφάλματός του μειώνεται κατά $1/(\sqrt{n})$, όπου n είναι ο αριθμός των αποδόσεων που υπολογίζονται κατά μέσο όρο.

Αλγοριθμος 2.2: *On-policy Monte Carlo control*

Initialise Q and π arbitrarily
Returns (s, a) ← empty list $\forall s \in S, a \in A$
repeat
    for $s \in S$ and $a \in A$ **do**
        Generate an episode using ε-greedy π starting with s, a
        for s, a in the episode **do**
            Returns(s, a) ← append return following s, a
            Q(s, a) = average(Returns(s, a))
        end for
        for s in the episode **do**
            $\pi(s) = argmax_a Q(s, a)$
        end for
    end for
until convergence

Στις μεθόδους εκτός πολιτικής έχουμε δύο πολιτικές :

- πολιτική-στόχος (target policy) είναι η πολιτική που μαθαίνεται και ειναι άπληστη σε σχέση με την Q
- πολιτική συμπεριφοράς (behaviour policy), η πολιτική που παράγει συμπεριφορά. Πρέπει να έχει μη μηδενική πιθανότητα επιλογής όλων των ενεργειών που μπορεί να επιλεγούν από την πολιτική στόχου (coverage). Για να το εξασφαλίσουμε αυτό απαιτούμε η πολιτική συμπεριφοράς να είναι μαλακή (δηλ, να επιλέγει όλες τις ενέργειες σε όλες τις καταστάσεις με μη μηδενική πιθανότητα).

Η πολιτική συμπεριφοράς μ μπορεί να είναι οποιαδήποτε, αλλά προκειμένου να διασφαλιστεί η σύγκλιση π στη βέλτιστη πολιτική, πρέπει να ληφθεί ένας άπειρος αριθμός από ανταμοιβές για κάθε ζεύγος κατάστασης και δράσης.

Αλγοριθμος 2.3: *Off-policy Monte Carlo control*

Initialise Q arbitrarily, $C(s,a) = 0 \forall s \in S, a \in A \pi \leftarrow$ greedy with respect to Q
repeat
  Generate an episode $[s_0, a_0, ..., a_{T-1}, s_T]$ using soft policy μ
  $R \leftarrow 0, W \leftarrow 1$
  for $t \leftarrow 0, T$ **do**
    $R \leftarrow \gamma R + r_{t+1}$
    $C(s_t, a_t) \leftarrow C(s_t, a_t) + W$
    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \frac{W}{C(s_t, a_t)}(R - Q(s_t, a_t))$
    $\pi(s) = argmax_a Q(s, a)$
    if $a_t \neq \pi(s_t)$ **then**
      Exit for loop
    end if
    $W \leftarrow W \frac{1}{\mu(a_t, s_t)})$
  end for
until convergence

2.3.2 Μάθηση προσωρινής διαφοράς

Οι μέθοδοι προσωρινής διαφοράς (Temporal-difference , TD) είναι παρόμοιες με τις μεθόδους δυναμικού προγραμματισμού, δηλαδή ενημερώνουν τις εκτιμήσεις που βασίζονται εν μέρει σε άλλες εκτιμήσεις, χωρίς να περιμένουν το τελικό αποτέλεσμα (bootstrap) άλλα και την μέθοδο Monte Carlo, μαθαίνουν απευθείας από την ακατέργαστη εμπειρία χωρίς μοντέλο της δυναμικής του περιβάλλοντος. Οι μέθοδοι TD περιμένουν μόνο μέχρι το επόμενο χρονικό βήμα για να ενημερώσουν τις εκτιμήσεις των τιμών. Τη χρονική στιγμή t + 1 σχηματίζουν αμέσως έναν στόχο και κάνουν μια ενημέρωση χρησιμοποιώντας την παρατηρούμενη ανταμοιβή r_t + 1 και την τρέχουσα εκτίμηση $V(s_t + 1)$. [20]

$$V(s_t) \leftarrow V(s_t) + a(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)) \tag{2.39}$$

, όπου α > 0 . Να σημειωθεί ότι αυτό είναι παρόμοιο με την ενημέρωση MC με τη διαφορά ότι πραγματοποιείται σε κάθε βήμα. Παρόμοια με τις μεθόδους DP, η μέθοδος TD βασίζει την ενημέρωσή της σε εν μέρει σε μια υπάρχουσα εκτίμηση - μια μέθοδο bootstrapping.Η παράμετρος α είναι το βήμα μάθησης (learning rate) του πράκτορα. Ο ορός :

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{2.40}$$

ονομάζετε σφάλμα εκτίμησής, το οποίο εμφανίζεται με διάφορες μορφές στην ενισχυτική μάθηση.Το σφάλμα TD σε κάθε χρονική στιγμή είναι το σφάλμα της εκτίμησης που έγινε εκείνη τη στιγμή. Επειδή το σφάλμα TD στο βήμα t εξαρτάται από την επόμενη κατάσταση και την επόμενη ανταμοιβή, δεν είναι πραγματικά διαθέσιμο μέχρι το βήμα t + 1. Η ενημέρωση της συνάρτησης αξίας με το σφάλμα TD ονομάζεται εφεδρικό σφάλμα. Το σφάλμα TD σχετίζεται με την εξίσωση Bellman.
[8], [21]

SARSA

Η SARSA είναι μια μέθοδος TD εντός της πολιτικής. Η οποία αποτελεί μια γενικευμένη προσεγγιστική μέθοδος πολιτικής. Ισορροπεί μεταξύ διερεύνησης και εκμετάλλευσης.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + a(r_t + 1 + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \tag{2.41}$$

Η ενημέρωση αυτή γίνεται μετά από κάθε μετάβαση από μια μη τερματική κατάσταση st . Εάν η s_{t+1} είναι τερματική, τότε το $Q(s_{t+1}, a_{t+1})$ ορίζεται ως μηδέν. Η εξίσωση ενημέρωσης για την SARSA εξαρτάται από την τρέχουσα κατάσταση, την τρέχουσα δράση, την ανταμοιβή που λαμβάνεται, την επόμενη κατάσταση και την επόμενη δράση. Αυτή η παρατήρηση οδήγησε στην ονομασία της τεχνικής μάθησης ως SARSA (State Action Reward State Action) που συμβολίζει την πλειάδα (s, a, r, s', a').

Αλγοριθμος 2.4: *SARSA*

Initialise Q arbitrarily, Q(terminal, ·) = 0
repeat
    Initialize s
    Choose a ε-greedily
    repeat
        Take action a, observe r , s'
        Choose $a' \varepsilon$-greedily
        $Q(s, a) \leftarrow Q(s, a) + a(r + \gamma Q(s', a') - Q(s, a))$
        $s \leftarrow s', a \leftarrow a'$
    until s is terminal
until convergence

Ο αλγόριθμος SARSA είναι ένας αλγόριθμος εντός πολιτικής, πράγμα που σημαίνει ότι κατά την εκμάθηση της βέλτιστης πολιτικής χρησιμοποιεί την τρέχουσα εκτίμηση της βέλτιστης πολιτικής για τη δημιουργία της συμπεριφοράς. Ο SARSA συγκλίνει σε μια βέλτιστη πολιτική αρκεί όλα τα ζεύγη κατάστασης-δράσης να επισκέπτονται άπειρες φορές και η πολιτική να συγκλίνει οριακά στην άπληστη πολιτική ($\varepsilon = 1/t$). [20]

Προσδοκώμενη Sarsa (Expected Sarsa)

Μια εναλλακτική λύση στη λήψη μιας τυχαίας ενέργειας και χρησιμοποιώντας την εκτίμηση της συνάρτησης Q για την ενέργεια αυτή στο σφάλμα TD- (όπως στο SARSA) είναι η χρήση της αναμενόμενης τιμής της συνάρτησης Q.

$$\begin{aligned} Q(s_t, a_t) \leftarrow Q(s_t, a_t) + a(E[Q(s_{t+1}, a_{t+1})|s_{t+1}] - Q(s_t, a_t)) = \\ Q(s_t, a_t) + a(r_{t+1} + \gamma \sum_{a}^{'} \pi(a'|s_{t+1}) Q(st + 1, a') - Q(s_t, a_t)) \end{aligned} \tag{2.42}$$

Αν και υπολογιστικά πιο πολύπλοκη, η μέθοδος αυτή έχει χαμηλότερη διακύμανση.

Επίσης γενικά αποδίδει καλύτερα και μπορεί να είναι είτε εντός είτε εκτός πολιτικής.

2.3.3 Q-learning

Η Q-learning είναι μια μέθοδος TD εκτός πολιτικής, στην οποία η μάθηση της συνάρτησης δράσης-αξίας, Q, άμεσα προσεγγίζει τη βέλτιστη συνάρτηση δράσης-αξίας, ανεξάρτητα από την ακολουθούμενη πολιτική.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + a(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)) \tag{2.43}$$

Αυτό απλοποιεί δραματικά την ανάλυση του αλγορίθμου και επιτρέπει την απόδειξη της πρώιμης σύγκλισης: το μόνο που απαιτείται για την ορθή σύγκλιση είναι όλα τα ζεύγη $\forall a, s$ να συνεχίζουν να ενημερώνονται. [26]

Αλγοριθμος 2.5: *Q-learning*

```
Initialise Q arbitrarily, Q(terminal, ·) = 0
repeat
    Initialize s
    repeat
        Choose a ε-greedily
        Take action a, observe r, s'
        Q(s, a) ← Q(s, a) + a(r + γ max'_a Q(s', a') − Q(s, a))
        s ← s'
    until s is terminal
until convergence
```

Προσαρμοσμένη Q-μάθηση

Στην προσαρμοσμένη Q-μάθηση (Fitted Q-learning) οι εμπειρίες συγκεντρώνονται σε ένα δεδομένο σύνολο δεδομένων D με τη μορφή πλειάδων < s, a, r, s' > όπου η κατάσταση στο επόμενο χρονικό βήμα s' προκύπτει από το T (s, a, ·) και η ανταμοιβή r δίνεται από την R(s, a, s'). Ο αλγόριθμος ξεκινά με κάποια τυχαία αρχικοποίηση του των Q-τιμών $Q(s, a; \theta_0)$ όπου το θ_0 αναφέρεται στις αρχικές παραμέτρους (συνήθως έτσι ώστε οι αρχικές τιμές Q να είναι σχετικά κοντά στο 0, ώστε να να αποφεύγεται η αργή μάθηση). Στη συνέχεια, μια προσέγγιση των τιμών Q στο k-οστή επανάληψη $Q(s, a; \theta_k)$ ενημερώνεται προς την συνάρτησή στόχου (target value)

$$Y_k^Q = r + \gamma \max_{a' \in A} Q(s', a'; \theta_k), \tag{2.44}$$

όπου το θ_k αναφέρεται σε ορισμένες παραμέτρους που καθορίζουν τις τιμές Q στην k-οστή επανάληψη. [27]

NFQ

Στη νευρωνικά προσαρμοσμένη μάθηση Q (Neural Fitted Q NFQ), η κατάσταση μπορεί να δοθεί ως είσοδος στο δίκτυο Q και να δοθεί μια διαφορετική έξοδος για κάθε μία από τις πιθανές ενέργειες. Αυτό έχει ως αποτέλεσμά μια αποτελεσματική δομή που έχει το πλεονέκτημα ότι επιτυγχάνεται ο υπολογισμός της $max_{a' \in A} Q(s', a'; \partial k)$ σε ένα μόνο πέρασμα στο νευρωνικό δίκτυο για ένα δεδομένο s'. Οι Q- τιμές παραμετροποιούνται με ένα νευρωνικό δίκτυο $Q(s, a; \partial_k)$ όπου οι παράμετροι ∂_k ενημερώνονται με στοχαστική κάθοδο βασισμένη στην κλίση (gradient descent) ελαχιστοποιώντας την τετραγωνική συνάρτησή της απώλειάς:

$$L_{DQN} = (Q(s, a; \partial_k) - Y_k^Q)^2 \tag{2.45}$$

Έτσι, η ενημέρωση Q-learning καταλήγει στην ενημέρωση των παραμέτρων:

$$\partial_{k+1} = \partial_k + a(Y_k^Q - Q(s, a; \partial_k))\nabla_{\partial_k} Q(s, a; \partial_k), \tag{2.46}$$

όπου το α είναι ένα κλιμακωτό μέγεθος βήματος που ονομάζεται ρυθμός μάθησης. Η χρήση της τετραγωνική απώλειας δεν είναι αυθαίρετη, εξασφαλίζει ότι το $Q(s, a; \partial_k)$ θα πρέπει να τείνει χωρίς προκατάληψη προς την αναμενόμενη τιμή της τυχαίας μεταβλητής Y_k^Q. Επομένως, εξασφαλίζει ότι η $Q(s, a; \partial_k)$ θα πρέπει να τείνει προς την Q*(s, a) μετά από πολλές επαναλήψεις, υποθέτοντας ότι το νευρωνικό δίκτυο είναι καλά προσαρμοσμένο για την εργασία και ότι η εμπειρία που έχει συγκεντρωθεί στο σύνολο δεδομένων D είναι επαρκής [28]

Κεφάλαιο 3

Βαθιά Ενισχυτική Μάθηση

3.1 Βαθιά Ενισχυτική Μάθηση

Η ενισχυτική μάθηση όπου η συνάρτηση αξίας, η πολιτική ή το μοντέλο, προσεγγίζεται μέσω ενός νευρωνικού δικτύου ονομάζεται βαθιά ενισχυτική μάθηση. Ποιο αναλυτικά λαμβάνουμε μεθόδους βαθιάς ενισχυτικής μάθησης (deep RL) όταν χρησιμοποιούμε βαθιά νευρωνικά δίκτυα για να αναπαραστήσουμε την κατάσταση ή την παρατήρηση και/ή για να προσεγγίσουμε οποιοδήποτε από τα ακόλουθα στοιχεία της ενισχυτικής μάθησης: συνάρτηση αξίας, $\hat{v}(s;\theta)$ ή $\hat{q}(s,a;\theta)$, πολιτική $\pi(a \mid s;\theta)$ και μοντέλο (συνάρτηση μετάβασης κατάστασης και συνάρτηση ανταμοιβής). Εδώ, οι παράμετροι θ είναι τα βάρη στα βαθιά νευρωνικά δίκτυα. Όταν χρησιμοποιούμε "ρηχά" μοντέλα, όπως γραμμική συνάρτηση, δέντρα απόφασης, κωδικοποίηση πλακιδίων, και ούτω καθεξής, ως προσέγγιση συνάρτησης, λαμβάνουμε "ρηχά" RL, και οι παράμετροι θ είναι οι παράμετροι βάρους σε αυτά τα μοντέλα. Η ευδιάκριτη διαφορά μεταξύ της βαθιάς RL και της "ρηχής" RL είναι το τι προσέγγιση συνάρτησης χρησιμοποιείται. Συνήθως χρησιμοποιούμε στοχαστική κατάβαση πλαγιάς (gradient descent) για την ενημέρωση των παραμέτρων βάρους στη βαθιά RL. Όταν συνδυάζονται εκτός πολιτικής, η προσέγγιση συνάρτησης, ιδίως η μη γραμμική προσέγγιση συνάρτησης, και η εκκίνησης, μπορεί να εμφανιστεί αστάθεια και απόκλιση. Ωστόσο, πρόσφατες εργασίες όπως το βαθύ δίκτυο Q το AlphaGo και άλλα σταθεροποιούν τη μάθηση και επιτυγχάνουν εξαιρετικά αποτελέσματα. [20]

3.2 DQN

Το Βαθύ Q-δίκτυο (Deep Q-Network,DQN) είναι ένας αλγόριθμος γενικού σκοπού που, συνδυάζει την ενισχυτική μάθηση με μια κατηγορία τεχνητών νευρωνικών δικτύων, γνωστά ως βαθιά νευρωνικά δίκτυα. Ο πράκτορας αλληλεπιδρά με ένα περιβάλλον μέσω μιας ακολουθίας παρατηρήσεων, ενεργειών και ανταμοιβών. Ο στόχος του πράκτορα είναι να επιλέξει ενέργειες με τρόπο που να μεγιστοποιεί τη συσσωρευτική μελλοντική ανταμοιβή. Πιο συγκεκριμένα, χρησιμοποιούμε ένα βαθύ συνελικτικό νευρωνικό δίκτυο για την προσέγγιση της βέλτιστης συνάρτησης αξίας-δράσης

$$Q*(s,a) = \max_{\pi} E[r_t + \gamma r_{r+1} + \gamma^2 r_{r+2} + ...|s_t = s, a_t = a, \pi] \tag{3.1}$$

το οποίο είναι το μέγιστο άθροισμα των ανταμοιβών r_t προ εξοφλημένων κατά γ σε κάθε

χρονικό βήμα t, που μπορεί να επιτευχθεί από μια πολιτική συμπεριφοράς π = P(a | s), αφού γίνει μια παρατήρηση (s) και εκτελώντας μια ενέργεια (a). [29]

Είναι γνωστό ότι η ενισχυτική μάθηση είναι ασταθής ή και αποκλίνει όταν χρησιμοποιείται μια η γραμμική προσέγγισή συναρτήσεων, όπως ένα νευρωνικό δίκτυο, για την αναπαράσταση της συνάρτησης Q. Αυτή η αστάθεια έχει διάφορες αιτίες: τις συσχετίσεις που υπάρχουν στην ακολουθία των παρατηρήσεων, το γεγονός ότι μικρές ενημερώσεις στην Q μπορεί να αλλάξουν σημαντικά την πολιτική και συνεπώς την κατανομή των δεδομένων, και τις συσχετίσεις μεταξύ των τιμών δράσης Q και των τιμών-στόχων $r + \gamma \max'_a Q(s', a)$. Η συνάρτησή στόχου που χρησιμοποιείται από το DQN είναι:

$$Y_t^{DQN} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_t^-) \tag{3.2}$$

Αντιμετωπίζουμε αυτές τις αστάθειες με μια νέα παραλλαγή της μάθησης Q, η οποία χρησιμοποιεί δύο βασικές ιδέες. Πρώτον, χρησιμοποιήσαμε έναν βιολογικά εμπνευσμένο μηχανισμό που ονομάζεται επανάληψη εμπειρίας που τυχαιοποιεί τα δεδομένα, αφαιρώντας έτσι τις συσχετίσεις στην ακολουθία παρατηρήσεων και εξομαλύνοντας τις αλλαγές στην κατανομή των δεδομένων . Δεύτερον, χρησιμοποιήσαμε μια επαναληπτική ενημέρωση που προσαρμόζει τις τιμές δράσης Q προς τις τιμές στόχου που ενημερώνονται μόνο περιοδικά, μειώνοντας έτσι τις συσχετίσεις με τον στόχο.

Ενώ υπάρχουν και άλλες σταθερές μέθοδοι για την εκπαίδευση νευρωνικών δικτύων στο πλαίσιο της ενισχυτικής μάθησης, όπως η NFQ που είδαμε στην ενότητά Q-learning , οι μέθοδοι αυτές περιλαμβάνουν την επαναλαμβανόμενη εκπαίδευση των δικτύων εκ νέου σε εκατοντάδες επαναλήψεις. Κατά συνέπεια, αυτές οι μέθοδοι, σε αντίθεση με τον αλγόριθμό μας, είναι πολύ αναποτελεσματικές για να χρησιμοποιηθούν με επιτυχία σε μεγάλα νευρωνικά δίκτυα. Παραμετροποιούμε μια προσεγγιστική συνάρτηση τιμής $Q(s, a; \theta_i)$ χρησιμοποιώντας ένα βαθύ συνελικτικό νευρωνικό δίκτυο, στο οποίο θ_i είναι οι παράμετροί (δηλαδή τα βάρη) του Q-δικτύου στην επανάληψη i. Αποθηκεύουμε τις εμπειρίες $e_t = (s_t, a_t, r_t, s_{t+1})$ του πράκτορα σε κάθε χρονικό βήμα t σε ένα σύνολο δεδομένων $D_t = e_1, ..., e_t$. Κατά τη διάρκεια της μάθησης, εφαρμόζουμε ενημερώσεις Q-learning, σε δείγματα (ή μίνι-πακέτα) εμπειριών $(s, a, r, s') \sim U(D)$, που αντλούνται ομοιόμορφα-τυχαία από τη δεξαμενή των αποθηκευμένων δειγμάτων. Η ενημέρωση Q-μάθησης στην επανάληψη i χρησιμοποιεί την ακόλουθη συνάρτηση απωλειών:

$$L_i(\theta_i) = E_{(s,a,r,s')\sim U(D)}[r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a, \theta_i))^2 \tag{3.3}$$

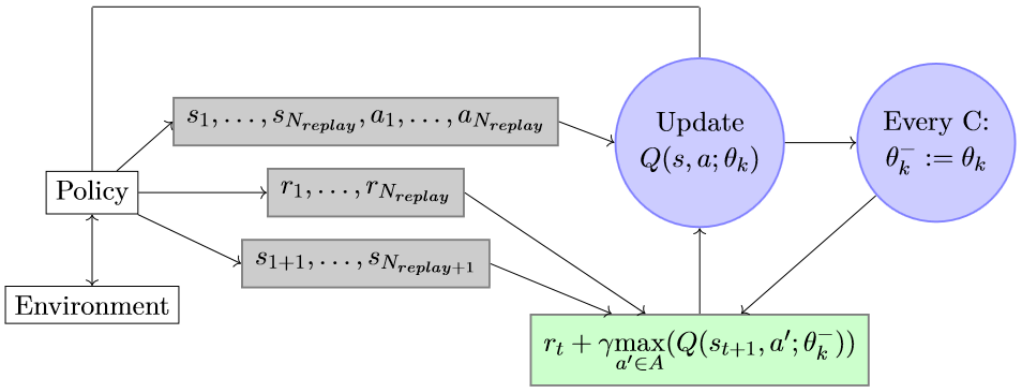
όπου γ είναι ο παράγοντας προεξόφλησης που καθορίζει τον ορίζοντα του πράκτορα, θ_i είναι οι παράμετροι του Q-δικτύου στην επανάληψη i και θ_i^- οι παράμετροι του δικτύου. [30] Οι παράμετροι του δικτύου-στόχου θ_i^- ενημερώνονται μόνο με τις παραμέτρους του Q-δικτύου θ_i κάθε $C \in N$ επαναλήψεις με την ακόλουθη ανάθεση: $\theta_k^- = \theta_k$. Αυτό αποτρέπει τη γρήγορη διάδοση των ασταθειών και μειώνει τον κίνδυνο απόκλισης.

Το ανάδελτα της συνάρτηση απωλειών, με σεβασμό στα βάρη, μας δίνει την ακόλουθη

κλίση:

$$\nabla_{\partial_i} L_i(\partial_i) = E_{s,a,r,s'}[(r + \gamma \max_{a'} Q(s', a'; \partial_i^-) - Q(s, a, \partial_i))\nabla_{\partial_i} Q(s, a, \partial_i)] \tag{3.4}$$

Να σημειωθεί ότι αυτός ο αλγόριθμος δεν έχει μοντέλο: επιλύει το έργο της ενισχυτικής μάθησης απευθείας χρησιμοποιώντας δείγματα από τον εξομοιωτή, χωρίς να εκτιμά ρητά την ανταμοιβή και τη δυναμική μετάβασης P(r,s'|s,a). Είναι επίσης εκτός πολιτικής: μαθαίνει για την άπληστη πολιτική $a = argmax_{a'} Q(s, a'; \partial)$, ενώ ακολουθεί μια συμπεριφορά που εξασφαλίζει επαρκή εξερεύνηση του χώρου καταστάσεων. Στην πράξη, η συμπεριφορα συχνά επιλέγεται από μια πολιτική ε-greedy, που ακολουθεί την άπληστη πολιτική με πιθανότητα 1-ε και επιλέγει μια τυχαία ενέργεια με πιθανότητα ε.



Σχήμα 3.1: *Σχεδιάγραμμα του αλγορίθμου DQN. Το Q(s, a; ∂_k) αρχικοποιείται σε τυχαίες τιμές (κοντά στο 0) παντού στο πεδίο του και η μνήμη αναπαραγωγής είναι αρχικά κενή. Οι παράμετροι στόχου του Q-δικτύου ∂_k^- ενημερώνουνται μόνο κάθε C επαναλήψεις με τις παραμέτρους του Q-δικτύου ∂_k και διατηρούνται σταθερές μεταξύ των ενημερώσεων- η ενημέρωση χρησιμοποιεί μια μίνι-ομάδα (π.χ. 32 στοιχεία) πλειάδων < s, a > που λαμβάνουνται τυχαία από τη μνήμη αναπαραγωγής μαζί με την αντίστοιχη μίνι-ομάδα τιμών στόχου για τις πλειάδες. [21]*

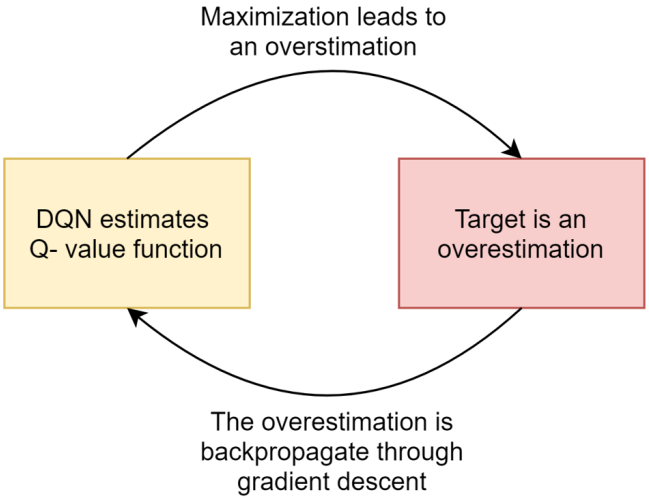
Αλγοριθμος 3.1: *Deep Q-learning with Experience Replay*

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights ϑ
Initialize target action-value function $\hat{Q}$ with weights $\partial^{-} = \partial$
for e **do**pisode = 1, M
    Initialize sequence $s_1 = x_1$ and preprocessed sequence $\varphi_1 = \varphi(s_1)$
    for t **do**=1,T
        With probability ε select a random action a_t otherwise select $a_t = argmax_a Q(\varphi(s_t), a; \partial)$
        Execute action a_t in emulator and observe reward r_t and image x_{t+1}
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\varphi_{t+1} = \varphi(s_{t+1})$
        Store transition $(\varphi_t, a_t, r_t, \varphi_{t+1})$ in D
        Sample random minibatch of transitions $(\varphi_j, a_j, r_j, \varphi_{j+1})$ from D
        Set $y_j = r_j$ for terminal $\varphi j + 1$ OR $y_j = r_j + \gamma max_{a'} Q(\varphi_{j+1}, a'; \partial)$ for non-terminal φ_{j+1}
        Perform a gradient descent step on $(y_j - Q(\varphi_j, a_j; \partial))^2$ with respect to the network parameters ϑ
        Every C steps reset $\hat{Q} = Q$
    end for
end for

[29]

Το πρόβλημά της υπερεκτίμησης

Ο αλγόριθμος Q-learning υπερεκτιμά τις τιμές δράσης υπό ορισμένες συνθήκες.



Σχήμα 3.2: *Απεικόνιση του τρόπου με τον οποίο ο DQN υπερεκτιμά την συνάρτηση Δράσης-Αξίας.*

Υπάρχουν 2 κύριοι λόγοι για την υπερεκτίμηση, όπως φαίνεται στην 3.2. Ο πρώτος λόγος οφείλεται στη συνάρτηση μεγιστοποίησης που χρησιμοποιείται για τον υπολογισμό της τιμής στόχου. Ας υποθέσουμε ότι οι πραγματικές άξιες δράσης συμβολίζονται με: $x(a_1)...x(a_n)$. Οι θορυβώδεις εκτιμήσεις που γίνονται από το DQN συμβολίζονται με:$Q(s, a_1; w), ...Q(s, a_n; w)$

$$\max_a Q(s, a, w) \geq \max_a(x(a)) \tag{3.5}$$

επομένως υπάρχει υπερεκτίμηση της πραγματικής τιμής της Q.

Ο δεύτερος λόγος είναι ότι οι υπερεκτιμημένες τιμές Q χρησιμοποιούνται και πάλι για την ενημέρωση των βαρών του δικτύου Q μέσω της οπισθοδιάδοσης. Αυτό καθιστά την υπερεκτίμηση πιο σοβαρή. Οι υπεραισιόδοξες εκτιμήσεις των τιμών δεν αποτελούν απαραίτητα πρόβλημα από μόνες τους. Εάν όλες οι τιμές ήταν ομοιόμορφα υψηλότερες, τότε οι σχετικές προτιμήσεις δράσης διατηρούνται και δεν θα περιμέναμε η προκύπτουσα πολιτική να είναι χειρότερη. Εάν, ωστόσο, οι υπερεκτιμήσεις δεν είναι ομοιόμορφες και δεν συγκεντρώνονται σε καταστάσεις για τις οποίες θέλουμε να μάθουμε περισσότερα, τότε μπορεί να επηρεάσουν αρνητικά την ποιότητα της πολιτικής που προκύπτει. Διαισθητικά όσο πιο συχνά εμφανίζεται ένα συγκεκριμένο ζεύγος κατάστασης, δράσης στο buffer αναπαραγωγής, τόσο μεγαλύτερη υπερεκτίμηση γίνεται σε αυτό το ζεύγος κατάστασης-δράσης.

3.2.1 Διπλό DQN

Μια προσαρμογή του αλγορίθμου DQN με στόχο να μειωθούν οι παρατηρούμενες υπερεκτιμήσεις αποτελεί το Διπλό Βαθύ Q-Δίκτυο (DDQN), το οποίο οδηγεί επίσης σε πολύ καλύτερη απόδοση σε διάφορα παιχνίδια. Για να μετριάσει την υπερεκτίμηση που προκαλείται από τη μεγιστοποίηση, το DDQN χρησιμοποιεί 2 Q-δίκτυα, ένα για τη λήψη ενεργειών και ένα άλλο για την ενημέρωση των βαρών μέσω οπισθοδιάδοσης.

Ο τελεστής max στην τυπική Q-learning και στην DQN, εξίσωσή 3.2 χρησιμοποιεί τις ίδιες τιμές τόσο για την επιλογή όσο και για την αξιολόγηση μιας ενέργειας. Αυτό καθιστά πιο πιθανό να επιλέξει υπερεκτιμημένες τιμές, με αποτέλεσμα υπεραισιόδοξες εκτιμήσεις Αξίας. Για να το αποτρέψουμε αυτό, μπορούμε να διαχωρίσουμε την επιλογή από την αξιολόγηση. Αυτή είναι η ιδέα πίσω από την Double Q-learning [31]. Στον αρχικό αλγόριθμο Double Q-learning, δύο συναρτήσεις άξιας μαθαίνουν αναθέτοντας σε κάθε εμπειρία τυχαία την ενημέρωση μιας από τις δύο συναρτήσεις αξίας, έτσι ώστε να υπάρχουν δύο σύνολα βαρών, θ και θ'. Για κάθε ενημέρωση, το ένα σύνολο βαρών χρησιμοποιείται για τον προσδιορισμό της άπληστης πολιτικής και το άλλο για τον προσδιορισμό της αξίας. Για μια σαφή σύγκριση, μπορούμε πρώτα να διαχωρίσουμε την επιλογή και την αξιολόγηση στην Q-learning και να ξαναγράψουμε τον στόχο της ως εξής:

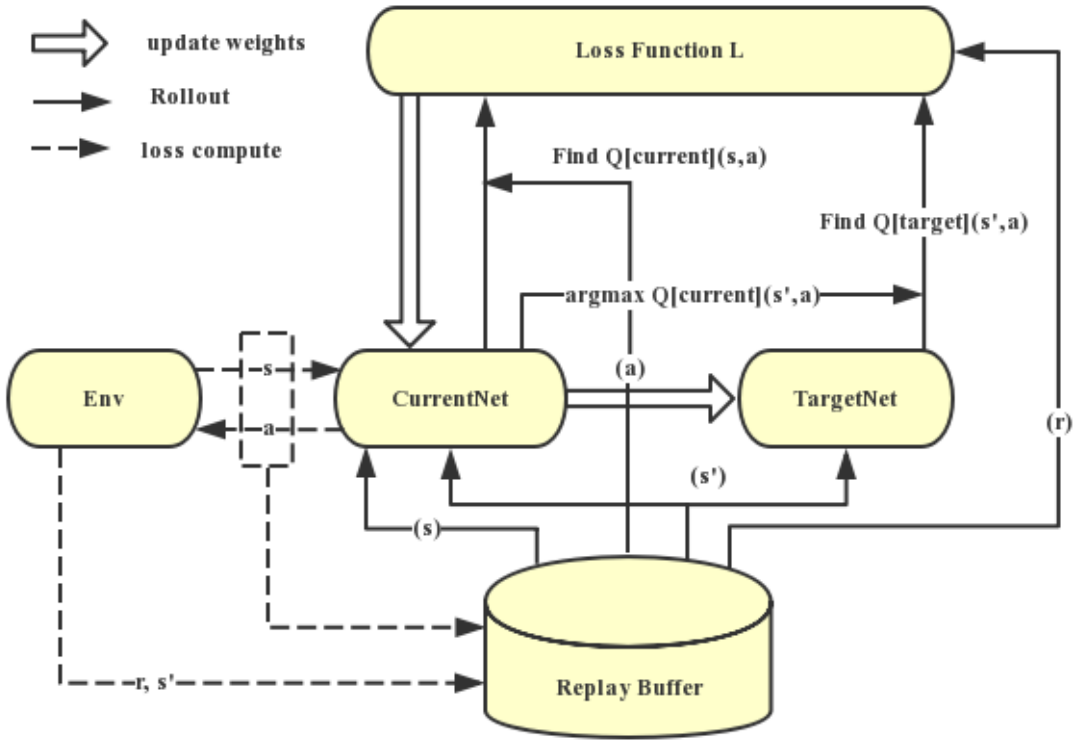
$$Y_t^Q = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta_t); \theta_t) \tag{3.6}$$

Το σφάλμα διπλής Q-μάθησης μπορεί τότε να γραφεί ως εξής:

$$Y_t^{DoubleQ} = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta_t); \theta'_t). \tag{3.7}$$

Να σημειώσουμε ότι η επιλογή της δράσης, στο argmax, εξακολουθεί να οφείλεται στα βάρη θ_t. Αυτό σημαίνει ότι, όπως και στην Q- μάθηση, εξακολουθούμε να εκτιμούμε την τιμή της άπληστης πολιτικής σύμφωνα με τις τρέχουσες αξίες, όπως ορίζονται από το θ_t. Ωστόσο, χρησιμοποιούμε το δεύτερο σύνολο βαρών θ'_t για να αξιολογήσουμε δίκαια την αξία αυτής της πολιτικής. Αυτό το δεύτερο σύνολο βαρών μπορεί να ενημερωθεί συμμετρικά,

αλλάζοντας τους ρόλους των θ και θ'. [31]



Σχήμα 3.3: *Αρχιτεκτονική DDQN. [32]*

3.3 Μέθοδοι κλίσης πολιτικής

Οι μέθοδοι πολιτικής κλίσης (policy gradient methods) αποτελούν μια ενδιαφέρουσα κατηγορία αλγορίθμων, καθώς εφαρμόζονται σε οποιαδήποτε μεταβλητή παραμετροποίηση πολιτικής- επεκτείνονται εύκολα στην προσέγγιση συναρτήσεων- ενσωματώνουν δομημένους χώρους καταστάσεων και δράσεων- και είναι εύκολο να να υλοποιηθούν στην προσομοίωση, χωρίς μοντέλα. Λόγω της ευελιξίας και της γενικότητάς τους, υπήρξε επίσης ένας καταιγισμός βελτιώσεων ώστε οι ιδέες αυτές να λειτουργήσουν εύρωστα με προσεγγίσεις που βασίζονται σε βαθιά νευρωνικά δίκτυα. Οι μέθοδοι αυτής της οικογένειας εκφράζουν μια πολιτική ως $\pi_\theta(a|s)$. Βελτιστοποιούν τις παραμέτρους θ είτε άμεσα μεγιστοποιώντας την απόδοση του στόχου $J(\pi_\theta)$, είτε έμμεσα, μεγιστοποιώντας τοπικές προσεγγίσεις της $J(\pi_\theta)$. Αυτή η βελτιστοποίηση πραγματοποιείται σχεδόν πάντα εντός-πολιτικής (on-policy), πράγμα που σημαίνει ότι κάθε ενημέρωση χρησιμοποιεί μόνο δεδομένα που συλλέγονται καθώς ενεργεί σύμφωνα με την πιο πρόσφατη έκδοση της πολιτικής. Η βελτιστοποίηση της πολιτικής περιλαμβάνει επίσης συνήθως μια προσέγγιση $V_\phi(s)$ για την συνάρτηση αξίας $V^\pi(s)$, η οποία χρησιμοποιείται για να βρεθεί ο τρόπος με τον οποίο θα ενημερωθεί η πολιτική. [33]

Αντικειμενική συνάρτηση

Ένας τρόπος μεγιστοποιήσουμε τις ανταμοιβές με βάση το θ, είναι να βρούμε μια αντικειμενική συνάρτηση J(θ) τέτοια ώστε

$$J(\theta) = V_{\pi_\theta}(s_0) \tag{3.8}$$

Όπου V_{π_θ} είναι η συνάρτηση αξίας για την πολιτική π_θ και s_0 είναι η αρχική κατάσταση. Άρα μεγιστοποίηση της $J(\theta)$σημαίνει μεγιστοποίηση της $V_{\pi_\theta}(s)$.Προκύπτει ότι :

$$\nabla J(\theta) = \nabla V_{\pi_\theta}(s_0) \tag{3.9}$$

Θεώρημα πολιτικής κλίσης

Ο υπολογισμός της κλίσης $\nabla_\theta J(\theta)$ είναι δύσκολος επειδή εξαρτάται τόσο από την επιλογή δράσης (που καθορίζεται άμεσα από την π_θ) όσο και από τη στάσιμη κατανομή των καταστάσεων ,δηλαδή την πιθανότητα να βρίσκεται κανείς στην κατάσταση s όταν ακολουθεί την πολιτική π_θ. Δεδομένου ότι το περιβάλλον είναι γενικά άγνωστο, είναι δύσκολο να εκτιμηθεί η επίδραση στην κατανομή καταστάσεων από μια ενημέρωση πολιτικής.

Το θεώρημα κλίσης πολιτικής παρέχει μια αναμόρφωση της παραγώγου της αντικειμενικής συνάρτησης ώστε να μην περιλαμβάνει την παράγωγο της κατανομής κατάστασης και να απλοποιεί πολύ τον υπολογισμό της κλίσης $\nabla_\theta J(\theta)$.

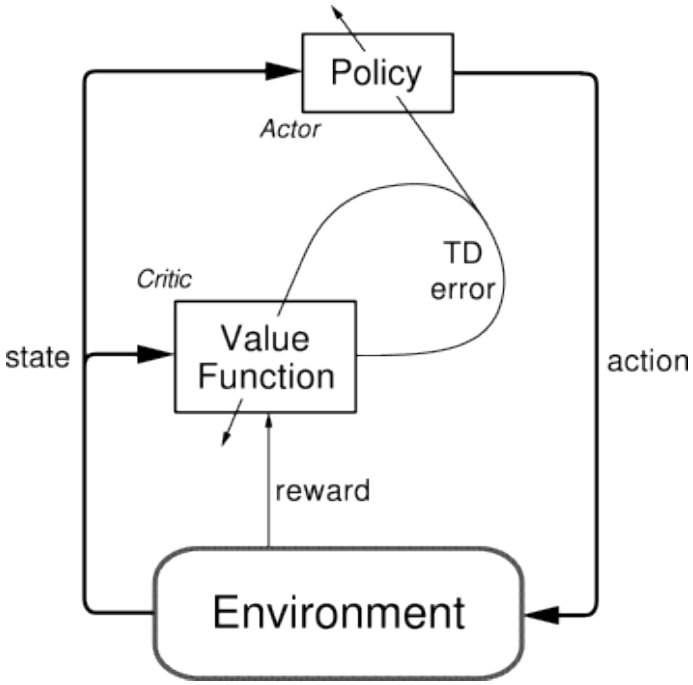
$$\nabla_\theta J(\theta) = \nabla_\theta \sum_{s\in\mathcal{S}} d^\pi(s) \sum_{a\in\mathcal{A}} Q^\pi(s,a)\pi_\theta(a|s) \propto \sum_{s\in\mathcal{S}} d^\pi(s) \sum_{a\in\mathcal{A}} Q^\pi(s,a)\nabla_\theta \pi_\theta(a|s) \tag{3.10}$$

Όπου $d^\pi(s)$ είναι η κατανομή υπό π (δηλαδή η πιθανότητα να βρίσκεται κανείς στην κατάσταση s όταν ακολουθεί την πολιτική π), Q(s,a) είναι η συνάρτηση αξίας δράσης και $\nabla_\theta \pi_\theta(a|s)$ ειναι η κλίση της π δεδομένης της s και της θ. Τέλος, $\propto$ σημαίνει αναλογικό.

Άρα το θεώρημα λέει ότι $\nabla_\theta J(\theta)$ είναι ανάλογο του αθροίσματος της συνάρτησης Q επί την κλίση των πολιτικών για όλες τις ενέργειες στις καταστάσεις στις οποίες μπορεί να βρισκόμαστε.

3.3.1 Δράστης κριτής

Οι μέθοδοι δράστη κριτή είναι μέθοδοι μάθησης χρονικών διαφορών (TD) που διαθέτουν μια ξεχωριστή δομή μνήμης για τη σαφή αναπαράσταση της πολιτικής ανεξάρτητα από τη συνάρτηση αξίας. Η δομή της πολιτικής είναι γνωστή ως δράστης, επειδή χρησιμοποιείται για την επιλογή ενεργειών, και η εκτιμώμενη συνάρτηση αξίας είναι γνωστή ως κριτής, επειδή επικρίνει τις ενέργειες που γίνονται από τον δράστη. Η μάθηση είναι πάντα επί της πολιτικής: ο κριτής πρέπει να μάθει και να επικρίνει οποιαδήποτε πολιτική ακολουθείται αυτή τη στιγμή από τον δράστη. Η κριτική παίρνει τη μορφή σφάλματος (TD). Αυτό το κλιμακωτό σήμα είναι η μοναδική έξοδος του κριτικού και οδηγεί όλη τη μάθηση τόσο στον δράστη όσο και στον κριτή. [20]



Σχήμα 3.4: *Η αρχιτεκτονική της μεθόδου δράστη-κριτή. [22]*

Συνήθως, ο κριτής είναι μια συνάρτηση κατάστασης-αξίας. Μετά από κάθε επιλογή δράσης, ο κριτής αξιολογεί τη νέα κατάσταση για να καθορίσει αν τα πράγματα πήγαν καλύτερα ή χειρότερα από το αναμενόμενο. Αυτή η αξιολόγηση είναι το σφάλμα (TD):

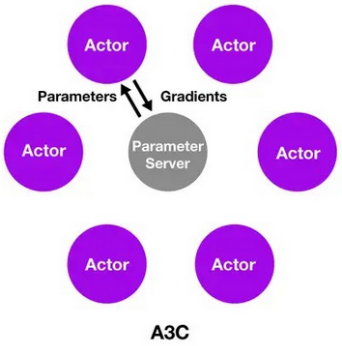
$$\delta_t = r_t + \gamma V(s_t + 1) - V(s_t) \tag{3.11}$$

όπου V είναι η τρέχουσα συνάρτηση αξίας που εφαρμόζεται από τον κριτή. Εάν το σφάλμα είναι θετικό, υποδηλώνει ότι η τάση επιλογής της a_t θα πρέπει να ενισχυθεί για το μέλλον, ενώ εάν το σφάλμα TD είναι αρνητικό, υποδηλώνει ότι η τάση θα πρέπει να αποδυναμωθεί. [21]

3.3.2 A3C

Ο αλγόριθμος κριτή δράστη ασύγχρονου πλεονεκτήματος (asynchronous advantage actor-critic (A3C)), χρησιμοποιεί πολλαπλούς πράκτορες με κάθε πράκτορα να έχει τις δικές του παραμέτρους δικτύου και ένα αντίγραφο του περιβάλλοντος. Αυτοί οι πράκτορες αλληλεπιδρούν με τα αντίστοιχα περιβάλλοντά τους ασύγχρονα, μαθαίνοντας με κάθε αλληλεπίδραση. Κάθε πράκτορας ελέγχεται από ένα ενιαίο δίκτυο. Καθώς κάθε πράκτορας αποκτά περισσότερη γνώση, συμβάλλει στη συνολική γνώση του συνολικού δικτύου. Η παρουσία ενός συνολικού δικτύου επιτρέπει σε κάθε πράκτορα να έχει πιο διαφοροποιημένα δεδομένα εκπαίδευσης.

Ο A3C διατηρεί μια πολιτική $\pi(a_t|s_t;\partial)$ και μια εκτίμηση της συνάρτησής άξιας $V(s_t;\partial_v)$. Όπως και η παραλλαγή της Q-learning n-βημάτων, χρησιμοποιεί το ίδιο μείγμα επιστροφών n-βήματος για την ενημέρωση τόσο της πολιτικής όσο και της συνάρτησης αξίας. Η πολιτική και η συνάρτηση άξιας ενημερώνονται μετά από κάθε t_{max} ενέργειες ή όταν φτάσουμε σε μια



Σχήμα 3.5: *Αρχιτεκτονική A3C που χρησιμοποιεί πολλαπλούς κατανεμημένους δράστες για την εκμάθηση των παραμέτρων του πράκτορα.*

τερματική κατάσταση. Η ενημέρωση που πραγματοποιείται από τον αλγόριθμο μπορεί να θεωρηθεί ως

$$\nabla_{\partial'} log\pi(a_t|s_t;\partial')A(s_t,a_t;\partial,\partial_v) \tag{3.12}$$

όπου $A(s_t,a_t;\partial,_\partial v)$ είναι μια εκτίμηση της συνάρτησης πλεονεκτήματος που δίνεται από τη σχέση

$$A(s_t,a_t;\partial,_\partial v) = \sum_{i=0}^{k-1} \gamma^i r_t + i + \gamma^k V(s_{t+k};\partial_v) - V(s_t;\partial_v), \tag{3.13}$$

όπου το k μπορεί να μεταβάλλεται από κατάσταση σε κατάσταση και περιορίζεται από t_{max}. [34]

Όπως και με τις μεθόδους βασισμένες στην αξία, βασιζόμαστε σε παράλληλους εκπαιδευόμενους δράστες και συσσωρευμένες ενημερώσεις για τη βελτίωση της σταθερότητας της εκπαίδευσης. Σημειώνουμε ότι ενώ οι παράμετροι ϑ της πολιτικής και ∂_v της συνάρτησης αξίας εμφανίζονται ως ξεχωριστές για λόγους γενικότητας, στην πράξη πάντα μοιραζόμαστε κάποιες από τις παραμέτρους. [35] Συνήθως χρησιμοποιούμε ένα συνελικτικό νευρωνικό δίκτυο που έχει μία έξοδο softmax για την πολιτική $\pi(a_t|s_t;\partial)$ και μία γραμμική έξοδο για τη συνάρτηση αξίας $V(s_t;\partial_v)$, με όλα τα επίπεδα μη εξόδου να είναι κοινά. Η κλίση της πλήρους αντικειμενικής συνάρτησης που περιλαμβάνει τον όρο ρύθμισης της εντροπίας ως προς τις παραμέτρους της πολιτικής έχει τη μορφή

$$\nabla_{\partial'} log\pi(a_t|s_t;\partial')(R_t - V(s_t;\partial_v)) + \beta\nabla_{\partial'} H(\pi(s_t;\partial')), \tag{3.14}$$

όπου Η είναι η εντροπία. Η υπερπαράμετρος β ελέγχει την ισχύ του όρου ρύθμισης της εντροπίας.

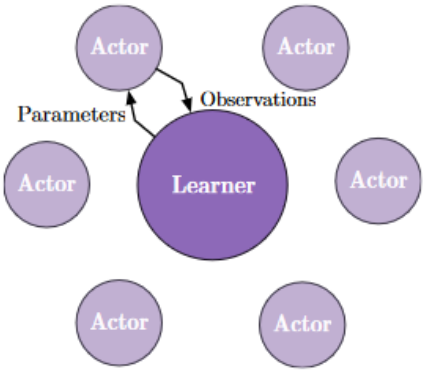
3.3.3 IMPALA

Η Αρχιτεκτονική σημαντικών βαρών δράστη-μαθητή (Importance Weighted Actor-Learner Architecture (IMPALA)) έχει τη δυνατότητα να εκτελείται σε χιλιάδες μηχανές χωρίς να θυσιάζεται η σταθερότητα της εκπαίδευσης ή αποδοτικότητα των δεδομένων. Σε αντίθεση με τους πράκτορες που βασίζονται στην A3C, στους οποίους οι εργαζόμενοι μεταδίδουν τις κλίσεις σε σχέση με τις παραμέτρους της πολιτικής σε έναν κεντρικό διακομιστή παραμέτρων, οι

Αλγοριθμος 3.2: *Asynchronous advantage actor-critic -ψευδοκώδικας για κάθε νήμα πράκτορα-μαθητή.*

// Assume global shared parameter vectors ϑ and ∂_v and global shared counter T = 0
// Assume thread-specific parameter vectors ϑ' and ∂'_v
Initialize thread step counter $t \leftarrow 1$
repeat
    Reset gradients: $d\partial \leftarrow 0$ and $d\partial_v \leftarrow 0$.
    Synchronize thread-specific parameters ϑ' = ϑ and $\partial'_v = \partial_v$
    $t_{start} = t$
    Get state s_t
    repeat
        Perform at according to policy $\pi(a_t|s_t;\partial')$
        Receive reward r_t and new state $s_t + 1$
        $t \leftarrow t + 1$
        $T \leftarrow T + 1$
    until terminal s_t OR $t - t_{start} == t_{max}$
    R = 0 for terminal s_t OR
    $R = V(st, \partial'_v)$ for non-terminal s_t // Bootstrap from last state
    for $i \in t - 1, ..., t_{start}$ **do**
        $R \leftarrow r_i + \gamma R$
        Accumulate gradients wrt ϑ':$d\partial \leftarrow d\partial + \nabla\partial' log\pi(ai|si;\partial')(R - V(s_i;\partial'_v))$
        Accumulate gradients wrt ∂'_v : $d\partial v \leftarrow d\partial_v + \partial(R - V(s_i;\partial'_v))2/\partial\partial'_v$
    end for
    Perform asynchronous update of ϑ using dϑ and of ∂_v using $d\partial_v$.
until $T > T_{max}$

πράκτορες IMPALA μεταδίδουν τις τροχιές της εμπειρίας (ακολουθίες καταστάσεων, δράσεις και ανταμοιβές) σε ένα κεντρικό μαθητή. Δεδομένου ότι ο μαθητής στο IMPALA έχει πρόσβαση σε πλήρεις τροχιές εμπειρίας, χρησιμοποιείτε μια GPU για να εκτελούνται ενημερώσεις σε μίνι-πακέτα τροχιών, ενώ ταυτόχρονα παραλληλίζουμε τις ανεξάρτητες από το χρόνο λειτουργίες. Ωστόσο, επειδή η πολιτική που χρησιμοποιείται για τη δημιουργία μιας τροχιάς μπορεί να υστερεί σε σχέση με την πολιτική του εκπαιδευόμενου κατά αρκετές ενημερώσεις τη στιγμή του υπολογισμού της κλίσης, η μάθηση γίνεται εκτός πολιτικής. Για να διορθωθεί η ασυμφωνία χρησιμοποιείτε ο αλγόριθμος V-trace off-policy actor-critic. [36]



Σχήμα 3.6: *Κάθε δράστης παράγει τροχιές και τις στέλνει μέσω μιας ουράς στον εκπαιδευόμενο. Πριν από την έναρξη της επόμενης τροχιάς, ο δράστης λαμβάνει τις τελευταίες παραμέτρους πολιτικής από τον μαθητή. [36]*

Το IMPALA χρησιμοποιεί μια δομή δράστη-κριτή για την εκμάθηση μιας πολιτικής π και μιας βασικής συνάρτησης V^{π} . Η διαδικασία παραγωγής εμπειριών είναι αποσυνδεδεμένη από την εκμάθηση των παραμέτρων των π και V^{π}. Η αρχιτεκτονική αποτελείται από ένα σύνολο δραστών, που παράγουν επανειλημμένα τροχιές εμπειριών, και έναν ή περισσότερους μαθητές, που χρησιμοποιούν τις εμπειρίες που αποστέλλονται από τους δράστες για να μάθουν το π εκτός πολιτικής. Στην αρχή κάθε τροχιάς, ένας δράστης ενημερώνει τη δική του τοπική πολιτική μ με την τελευταία πολιτική π του μαθητή και την εκτελεί για n βήματα στο περιβάλλον του. Μετά από n βήματα, ο δράστης στέλνει την τροχιά των καταστάσεων, των ενεργειών και των ανταμοιβών $x_1, a_1, r_1, ..., x_n, a_n, r_n$ μαζί με τις αντίστοιχες κατανομές πολιτικής $\mu(a_t|x_t)$ και την αρχική κατάσταση LSTM στον μαθητή μέσω μιας ουράς. Στη συνέχεια, ο μαθητής ενημερώνει την πολιτική του π σε παρτίδες τροχιών, κάθε μία από τις οποίες προέρχεται από πολλούς φορείς. Αυτή η απλή αρχιτεκτονική επιτρέπει την επιτάχυνση του μαθητή (ή των μαθητών) με τη χρήση GPU και την εύκολη κατανομή των δραστών σε πολλές μηχανές. Ωστόσο, η πολιτική π του μαθητή είναι ενδεχομένως αρκετές ενημερώσεις μπροστά από την πολιτική μ του δράστη κατά τη στιγμή της ενημέρωσης, επομένως υπάρχει καθυστέρηση πολιτικής μεταξύ των δραστών και του/των μαθητή/ων. Το V-trace διορθώνει αυτή την καθυστέρηση για να επιτύχει εξαιρετικά υψηλή απόδοση δεδομένων, διατηρώντας παράλληλα την αποδοτικότητα των δεδομένων.

Ο στόχος V-trace υπολογίζεται για κάθε δράστη που ακολουθεί μια πολιτική συμπεριφοράς μ και παράγει μια τροχιά (xt, at, rt).Ο στόχος V-trace n βημάτων για την προσέγγιση

της τιμής $V(x_s)$ είναι

$$v_s = V(x_s) + \sum_{t=s}^{s+n-1} \gamma^{t-s}(\prod_{i=s}^{t-1} c_i)\delta_t V \tag{3.15}$$

,οπου $\delta_t V$ η χρονική διαφορά

$$\delta_t V = \rho_t * (r_t + \gamma V(x_{t+1}) - V(x_t)) \tag{3.16}$$

Ο λόγος του στόχου και της πολιτικής του πράκτορα ορίζεται από τη σχέση $R_i = \frac{\pi(a_i|x_i)}{\mu(a_i|x_i)}$, που χρησιμοποιείται για τη δειγματοληψία σπουδαιότητας (importance sampling (IS))

$$\rho_i = \min(\bar{\rho}, R_i) \tag{3.17}$$

$$c_i = \min(\bar{c}, R_i) \tag{3.18}$$

Στην περίπτωση εντός-πολιτικής (όταν π = μ), και υποθέτοντας ότι $\hat{c} \geq 1$, τότε όλα τα $c_i = 1$ και $\rho_t = 1$, άρα η πάνω γράφετε ως:

$$v_s = V(x_s) + \sum_{t=s}^{s+n-1} \gamma^{t-s}(r_t + \gamma V(x_{t+1}) - V(x_t)) \tag{3.19}$$

$$= \sum_{t=s}^{s+n-1} \gamma^{t-s} r_t + \gamma^n V(x_{s+n}) \tag{3.20}$$

ο οποίος είναι ο εντός πολιτικής στόχος Bellman για n βήματα. Έτσι, στην περίπτωση της on-policy, το V-trace ανάγεται στην ενημέρωση Bellman n-βήματος για την on-policy. Αυτή η ιδιότητα επιτρέπει τη χρήση του ίδιου αλγορίθμου για δεδομένα εκτός και εντός πολιτικής. [36]

3.3.4 PPO

Ο αλγόριθμος βελτιστοποίησης κοντινής πολίτικης PPO (proximal policy optimization) αποτελεί μια απλούστευσή του TRPO(trust region policy optimization) με συγκρίσιμη αν όχι καλύτερή επίδοσή. Στοχεύει στο πώς μπορούμε να κάνουμε το μεγαλύτερο δυνατό βήμα ως προς την βελτίωση μιας πολιτικής χρησιμοποιώντας τα δεδομένα που διαθέτουμε αυτήν τη στιγμή, χωρίς η αλλαγή να είναι τόσο μεγάλη ώστε να καταρρεύσει η απόδοση μας. Άρα διατηρεί την νέα πολιτική κοντά στην παλιά. Ο αλγόριθμος PPO είναι ένας αλγόριθμος εντός πολιτικής και μπορεί να χρησιμοποιηθεί για περιβάλλοντα με διακριτή ή συνεχής δράση. [37], [20]

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{3.21}$$

Ο όρος π_θ είναι η πολιτική μας, είναι ένα νευρωνικό δίκτυο που παίρνει τις παρατηρούμενες καταστάσεις s_t από το περιβάλλον ως είσοδο και προτείνει ως έξοδο τις ενέργειες που πρέπει να γίνουν.

Το $r_t(\theta)$ υποδηλώνει τον λόγο πιθανοτήτων μεταξύ των νέων ενημερωμένων εξόδων πολι-

τικής και των εξόδων της προηγούμενης παλιάς έκδοσης του δικτύου πολιτικής, λαμβάνοντας υπόψη μια ακολουθία ενεργειών δειγματοληψίας.Η τιμή $r_t(\partial)$ θα είναι μεγαλύτερη από 1 εάν η ενέργεια είναι πιο πιθανή τώρα από ό,τι ήταν στην παλιά έκδοση της πολιτικής και θα είναι κάπου μεταξύ 0 και 1 εάν η ενέργεια είναι λιγότερο πιθανή τώρα, από ό,τι πριν από το τελευταίο βήμα διαβάθμισης (gradient step) [38]

Ο PPO-clip ενημερώνει τις πολιτικές μέσω :

$$\partial_{k+1} = \arg\max_{\partial} \mathop{\mathrm{E}}_{s,a\sim\pi_{\partial_k}} [L(s, a, \partial_k, \partial)] \tag{3.22}$$

,

συνήθως με πολλαπλά βήματα για τη μεγιστοποίηση του στόχου. Η αντικειμενική συνάρτησή $\mathcal{L}$ δίνεται από τη σχέση :

$$L^{CLIP}(\partial) = \hat{E}_t[min(r_t(\partial)\hat{A}_t), clip(r_t(\partial), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{3.23}$$

Η προσδοκία $\hat{E}t[....]$ υποδηλώνει τον εμπειρικό μέσο όρο για μια πεπερασμένη παρτίδα δειγμάτων, σε ένα αλγόριθμο που εναλλάσσεται μεταξύ δειγματοληψίας και βελτιστοποίησης.Ο όρος A_t είναι η εκτίμηση της συνάρτησης πλεονεκτήματος στο χρονικό βήμα t. Η ε είναι μια υπερπαράμετρος με μικρή τιμή που ελέγχει πόσο μακριά επιτρέπεται να πάει η νέα πολιτική από την παλιά.

Η αντικειμενική συνάρτηση που βελτιστοποιεί το PPO είναι ένας τελεστής προσδοκίας, επομένως αυτό σημαίνει ότι θα τον υπολογίσουμε σε παρτίδες τροχιών. Ο τελεστής προσδοκίας λαμβάνεται για τουλάχιστον δύο όρους, ο πρώτος από αυτούς τους όρους είναι $r_t(\partial)$ επί την εκτίμηση του πλεονεκτήματος, επομένως αυτός είναι ο προεπιλεγμένος στόχος για κανονικές κλίσεις πολιτικής (policy gradients) που ωθεί την πολιτική προς ενέργειες που αποφέρουν υψηλό θετικό πλεονέκτημα.Ο δεύτερος όρος είναι πολύ παρόμοιος με τον πρώτο εκτός από το ότι περιέχει μια περικομμένη έκδοση της αναλογίας $r_t(\partial)$ εφαρμόζοντας μια λειτουργία αποκοπής μεταξύ 1 μείον έψιλον και 1 συν έψιλον όπου το έψιλον είναι συνήθως περίπου 0,2. Τέλος, εφαρμόζεται ο τελεστής min στους δύο όρους για να λάβουμε το τελικό αποτέλεσμα .

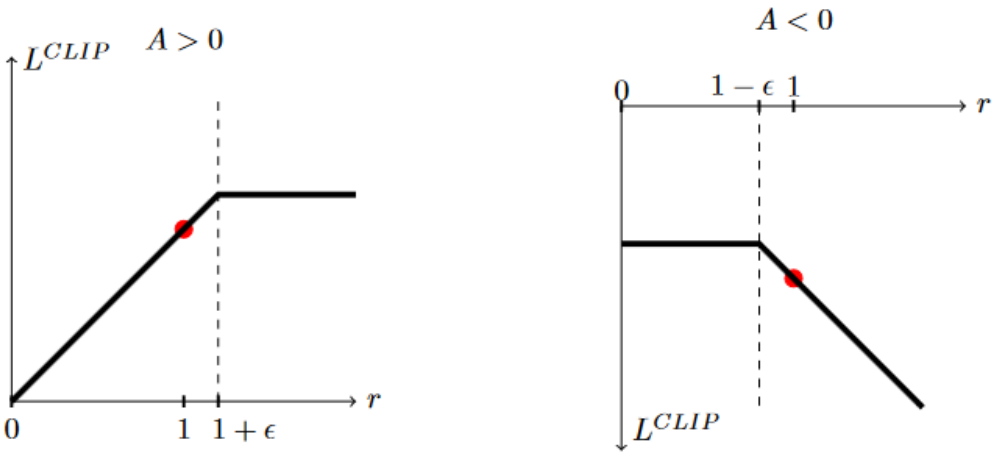
Για να το καταλάβουμε διαισθητικά, ας δούμε ένα απλό ζεύγος κατάστασης-δράσης (s,a) και ας σκεφτούμε τις περιπτώσεις.

Αριστερά το πλεονέκτημα είναι θετικό: Στην περίπτωση αυτή η συνεισφορά του στον στόχο μειώνεται σε

$$L(s, a, \partial_k, \partial) = \min\left(\frac{\pi_\partial(a|s)}{\pi_{\partial_k}(a|s)}, (1+\epsilon)\right) A^{\pi_{\partial_k}}(s, a). \tag{3.24}$$

Επειδή το πλεονέκτημα είναι θετικό, η αντικειμενική θα αυξηθεί αν η δράση γίνει πιο πιθανή - δηλαδή αν η $\pi_\partial(a|s)$ αυξηθεί. Αλλά το min σε αυτόν τον όρο θέτει ένα όριο στο πόσο μπορεί να αυξηθεί η αντικειμενική. Μόλις $\pi_\partial(a|s) > (1+\epsilon)\pi_{\partial_k}(a|s)$, το min μπαίνει σε λειτουργία και ο όρος αυτός φτάνει σε ένα ανώτατο όριο $(1+\epsilon)A^{\pi_{\partial_k}}(s, a)$. Συνεπώς: η νέα πολιτική δεν ωφελείται από την απομάκρυνση από την παλιά πολιτική.

Δεξιά το πλεονέκτημα για το συγκεκριμένο ζεύγος κατάστασης-δράσης είναι αρνητικό,



Σχήμα 3.7: *Τα διαγράμματα που παρουσιάζουν συνάρτησης L^{CLIP} ως συνάρτηση του λόγου πιθανοτήτων r, για θετικά πλεονεκτήματα (αριστερά) και αρνητικά πλεονεκτήματα (δεξιά). Ο κόκκινος κύκλος σε κάθε διάγραμμα δείχνει το σημείο εκκίνησης της βελτιστοποίησης, δηλαδή $r = 1$.*

οπότε η συνεισφορά του στον στόχο μειώνεται σε

$$L(s, a, \theta_k, \theta) = \max\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, (1-\epsilon)\right) A^{\pi_{\theta_k}}(s, a). \tag{3.25}$$

Επειδή το πλεονέκτημα είναι αρνητικό, ο στόχος θα αυξηθεί αν η δράση γίνει λιγότερο πιθανή - δηλαδή αν η $\pi_\theta(a|s)$ μειωθεί. Αλλά το max σε αυτόν τον όρο θέτει ένα όριο στο πόσο μπορεί να αυξηθεί η αντικειμενική . Μόλις $\pi_\theta(a|s) < (1-\epsilon)\pi_{\theta_k}(a|s)$, το max μπαίνει σε λειτουργία και ο όρος αυτός φτάνει σε ένα ανώτατο όριο $(1-\epsilon)A^{\pi_{\theta_k}}(s, a)$. Συνεπώς, και πάλι: η νέα πολιτική δεν ωφελείται από την απομάκρυνση από την παλιά πολιτική. [38]

Αυτό που είδαμε μέχρι τώρα είναι ότι η αποκοπή χρησιμεύει ως ρυθμιστής, αφαιρώντας τα κίνητρα για δραματική αλλαγή της πολιτικής, και η υπερπαράμετρος ϵ αντιστοιχεί στο πόσο μακριά μπορεί να απομακρυνθεί η νέα πολιτική από την παλιά, ενώ εξακολουθεί να επωφελείται η αντικειμενική.

Αλγοριθμος 3.3: *Algorithm 5 PPO with Clipped Objective*

1: Input: initial policy parameters θ_0, clipping threshold ε
2: **for** k = 0, 1, 2, ... **do**
3: Collect set of partial trajectories D_k on policy $\pi_k = \pi(\theta_k)$
4: Compute rewards-to-go $\hat{R}_t$
5: Estimate advantages $\hat{A}_t^{\pi k}$ using any advantage estimation algorithm
6: Compute policy update $\theta_{k+1} = \arg\max_\theta L_{\theta_k}^{CLIP}(\theta)$ by taking K steps of minibatch SGD (via Adam), where

$$L_{\theta_k}^{CLIP}(\theta) = E_{\tau\sim\pi k}[\sum_{t=0}^{T}[\min(r_t(\theta)\hat{A}_t^{\pi k}, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t^{\pi k})]] \tag{3.26}$$

7: **end for**

Κεφάλαιο 4

Περιγραφή Συστήματος

4.1 Περιγραφή παιχνιδιού

Στο παιχνίδι ελέγχουμε τον Mario και ο στόχος είναι να τερματίσει κάθε επίπεδο, στο Βασίλειο των Μανιταριών (Mushroom Kingdom) , και να σώσει την Princess Peach (Πριγκίπισσα Peach) από τον Bowser, πετυχαίνοντας το καλύτερο σκορ. Αυτό επιτυγχάνεται μειώνοντας τον χρόνο τερματισμού σκοτώνοντας εχθρούς και συλλέγοντας αντικείμενα. Ο παίκτης μετακινείται από την αριστερή μεριά της οθόνης στη δεξιά μεριά για να φτάσει τον πόλο της σημαίας στο τέλος κάθε επιπέδου.

Χρησιμοποιήσαμε το εξής περιβάλλον εκπαίδευσης gym-super-mario-bros κάνοντας τις κατάλληλες τροποποιήσεις. Είναι ένα OpenAI Gym περιβαλλον για το Super Mario Bros. και το Super Mario Bros. 2 (Lost Levels) στο Nintendo Entertainment System (NES) που χρησιμοποιεί τον εξομοιωτή nes-py.Το Gym είναι μια βιβλιοθήκη ανοικτού κώδικα σε Python για την ανάπτυξη και τη σύγκριση αλγορίθμων ενισχυτικής μάθησης, παρέχοντας μια τυποποιημένη διεπαφή προγραμματισμού εφαρμογών (API) για την επικοινωνία μεταξύ αλγορίθμων μάθησης και περιβαλλόντων, καθώς και ένα τυποποιημένο σύνολο περιβαλλόντων συμβατών με αυτό το API. Το περιβάλλον αυτό στέλνει μόνο τα καρέ (frames) από το κάθε επίπεδο του παιχνιδιού. Δεν αποστέλλονται τα menu επιλογών, εικόνες φόρτωσης κ.λ.π. [39] Το περιβάλλον τρέχει με 30 καρέ ανά δευτερόλεπτο (FPS), δέχεται μια ενέργεια και παράγει μια έγχρωμη εικόνα $240 \times 256 \times 3$ (από την οποία παράγεται μια παρατήρηση σε κάθε βήμα.

4.2 Σύνολο ενεργειών

Στο αρχικό παιχνίδι, κάθε ενέργεια που μπορεί να κάνει ο Mario εκτελείται χρησιμοποιώντας το χειριστήριο, πατώντας τα κουμπιά "A" και "B" και τον σταυρό κατευθύνσεων σε συνδυασμό ή μεμονωμένα. Στο περιβάλλον προσομοίωσης , τα κουμπιά συνδέονται με λέξεις-κλειδιά που περιγράφονται ως τέσσερις κατευθύνσεις, "A", "B" και "NOOP" όταν δεν κάνουμε καμία δράση .

Προσπαθώντας να κάνουμε τον λιγότερο χρόνο τερματισμού σε κάθε επίπεδο , η παραμονή σε ένα σημείο είναι η λιγότερο χρήσιμη ενέργεια καθώς χάνουμε χρόνο. Επομένως, καταργούμε τις ενέργειες που καθυστερούν την ολοκλήρωση του επιπέδου.

Οι διαθέσιμες ενέργειες είναι:

- NOOP: μην κάνεις τίποτα

- A: άλμα
- Up: σκύψε , μπες σε ένα σωλήνα ή κατέβα προς τα κάτω σε ένα “beanstalk” βλαστό φασολιού (σκάλα)
- Down: σκαρφάλωσε προς τα πάνω σε ένα βλαστό φασολιού (σκάλα)
- Right: μετακινήσου προς τα δεξιά
- Right + A: πήδηξε ή κολύμπησε προς τα πάνω ενώ κινείσαι προς τα δεξιά
- Right + B: τρέξε ή ρίξε μπάλες φωτιάς (μόνο όταν ο Mario βρίσκεται σε κατάσταση "Fireball") ενώ κινείται προς τα δεξιά
- Right + A + B: συνδυασμός των δύο παραπάνω

Υπάρχουν επίσης οι αντίστοιχες τέσσερις τελευταίες ενέργειες προς τα αριστερά. Αρά συνολικά έχουμε 12 διαθέσιμες δράσεις.

Το περιβάλλον εκπαίδευσής καθορίζει ένα προσαρμοσμένο σύνολο ενεργειών που μπορεί να χρησιμοποιήσει ο Mario με διάφορους βαθμούς πολυπλοκότητας. Η επιλογή ενός απλούστερου χώρου ενεργειών κάνει πιο γρήγορη και εύκολη την εκμάθηση του Mario, αλλά τον εμποδίζει να δοκιμάσει πιο σύνθετες κινήσεις που μπορεί να περιλαμβάνουν την είσοδο σε σωλήνες και την πραγματοποίηση προηγμένων αλμάτων που μπορεί να απαιτούνται για την επίλυση κάποιων επιπέδων. Υποστηρίζονται οι ακόλουθες επιλογές:

Right only

Ο Mario μπορεί ουσιαστικά να πηγαίνει μόνο προς τα δεξιά. Αυτό απλοποιεί τη διαδικασία εκπαίδευσης, αλλά εμποδίζει τον Mario να δοκιμάσει πιο σύνθετες ενέργειες. Υποστηρίζονται τα ακόλουθα κουμπιά:

• NOOP • Right • Right + A • Right + B • Right + A + B

Simple movement

Εκτός από την κίνηση προς τα δεξιά και το τρέξιμο/πήδημα, ο Mario μπορεί τώρα να περπατάει αριστερά και να πηδάει στη θέση του. Υποστηρίζονται τα ακόλουθα κουμπιά:

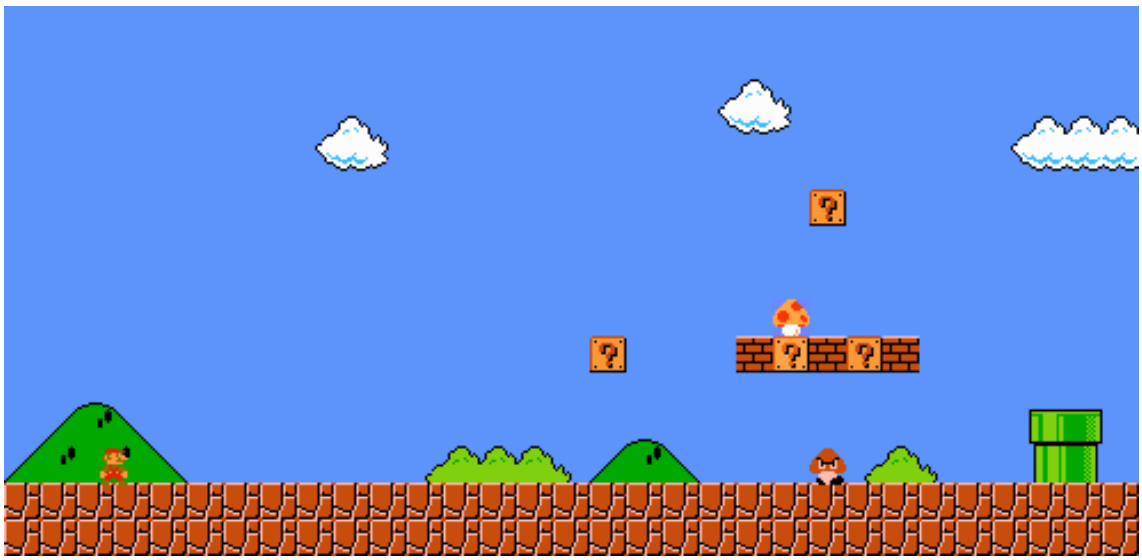
• NOOP • Right • Right + A • Right + B • Right + A + B • A • Left

Complex movement

Αυτή η ενέργεια επιτρέπει στον Mario να δοκιμάσει σχεδόν οποιαδήποτε από τις πιθανές ενέργειες του παιχνιδιού. Αυτή η επιλογή θα πρέπει να επιλέγεται από προεπιλογή για την πιο ρεαλιστική εξερεύνηση ενός επιπέδου, αλλά μπορεί να αυξήσει το χρόνο και την πολυπλοκότητα της εκμάθησης ενός επιπέδου. Αυτός είναι ο μόνος παρεχόμενος χώρος δράσης που επιτρέπει στον Mario να εισέλθει σε σωλήνες με κάθετο προσανατολισμό. Υποστηρίζονται τα ακόλουθα κουμπιά:

• NOOP • Right • Right + A • Right + B • Right + A + B • A • Left • Left + A • Left + B • Left + A + B • Down • Up

4.3 Περιβάλλον εκπαίδευσης



Σχήμα 4.1: *Στιγμιότυπο του παιχνιδιού*

Ως παιχνίδι κονσόλας , τα επίπεδα είναι το θεμέλιο όπου κινείται ο ελεγχόμενος χαρακτήρας. Σε αυτήν την εργασία , θεωρούμε ότι υπάρχουν 4 τύποι στοιχείων πλατφόρμας και 4 τύποι αντικειμένων σε κάθε επίπεδο. Οι πλατφόρμες είναι (Ground, Pit, Cannon και Pipe) και τα αντικείμενα (κέρμα, μανιτάρι, τούβλο, λουλούδι). Αναλυτικά :

- Ground: το έδαφός ,η βάση κάθε επιπέδου
- Pit: μια τρύπα στο έδαφος
- Cannon: ένα οδόφραγμα που πυροβολεί στην ευθεία έναν εχθρό που ονομάζεται "Bullet Bill" ανά σύντομα χρονικά διαστήματα
- Pipe: ένα κανάλι που μεταφέρει τον Mario σε άλλο σημείο του επιπέδου, θα το αγνοήσουμε στην παρούσα εργασία.

Τις περισσότερες φορές, ο Mario τρέχει στο έδαφος και πηδά για να αποφύγει τις τρύπες. Αν πέσει σε τρύπα τότε πεθαίνει. Το κανόνι (Cannon) λειτουργεί σχεδόν το ίδιο με ένα κανονικό εμπόδιο το οποίο μπορεί να αγγίξει ο Mario, ωστόσο παράγει εχθρούς περιοδικά.

Τα αντικείμενα είναι:

- Νόμισμα: κερδίζει πόντους που αυξάνουν το σκορ
- Μανιτάρι: αλλάζει τον Μάριο από "μικρό" σε "ψηλό" και κερδίζει πόντους
- Λουλούδι: αλλάζει τον Μάριο από "μικρό" ή "ψηλό" σε " φλογερό" και κερδίζει πόντους
- Τούβλο: μπορεί να δημιουργήσει τα παραπάνω 3 αντικείμενα όταν το σπάσει ή όταν το χτυπήσει ο Μάριο.

Σε αντίθεση με τα στοιχεία της πλατφόρμας, τα αντικείμενα μπορούν να εξαφανιστούν από τη σκηνή αν αλληλοεπιδράσουν με τον παίκτη. Τα νομίσματα, τα μανιτάρια και τα λουλούδια, εξαφανίζονται όταν τα καταναλώνει ο Μάριο. Ένα τούβλο, ανεξάρτητα από τον τύπο του, μπορεί επίσης να σπάσει και να εξαφανιστεί, εάν ο Μάριο δεν είναι στην κατάσταση “μικρός”. Ωστόσο, δεν είναι όλα τα τούβλα εύθραυστα, τα τούβλα που σπάνε είναι ακριβώς τα ίδια με τα άθραυστα μέχρι να χτυπηθούν .

Όταν ο Μάριο είναι "ψηλός", έχει διπλάσιο μέγεθος από ένα «μικρό» Μάριο. Όταν είναι " φλογερός ", μπορεί να ρίξει μέχρι δυο μπάλες φωτιάς τη φορά που σκοτώνουν εχθρούς όταν τους ακουμπήσουν. Ο Μάριο δε θα αλλάξει ξανά κατάσταση εάν καταναλώσει μανιτάρι και είναι ήδη ψηλός αντίστοιχα δε θα αλλάξει κατάσταση αν είναι "φλογερός" και καταναλώσει ένα λουλούδι. Κάθε φορά που έρχεται σε επαφή με εχθρούς κατεβαίνει μια κατάσταση κάτω με την εξής σειρά από " φλογερός "σε "ψηλός" και από κει σε "μικρός".

Υπάρχει ένα χαρακτηριστικό στο Super Mario Bros που ονομάζεται σωλήνες- τηλεμεταφορας (warp pipes). Ο σωλήνας-τηλεμεταφοράς επιτρέπει σε έναν παίκτη που μπαίνει μέσα να παραλείψει συγκεκριμένα κομμάτια του παιχνιδιού . Εάν ένας πράκτορας χρησιμοποιήσει τον σωλήνα- τηλεμεταφοράς, αυτό μπορεί να μειώσει σημαντικά το χρόνο για να κερδίσει ολόκληρο το παιχνίδι παρακάμπτοντας κάποια στάδια. Η χρήση αυτών επιτρέπει σε έναν πράκτορα να ανταγωνίζεται τους ανθρώπινους παίκτες ή ακόμα και να ανακαλύπτει περισσότερους αποτελεσματικός συνδυασμός για να νικήσει το παιχνίδι. Ωστόσο, στο εκπαιδευτικό μας περιβάλλον, αγνοούμε τη δυνατότητα της τηλεμεταφοράς μέσο των warp pipes για απλοποίηση του προβλήματος.

Καθώς ο πράκτορας κινείται σε διαφορετικές θέσεις σε ένα επίπεδο, οι μεταβάσεις δημιουργούνται από τις αλληλεπιδράσεις του με το περιβάλλον. Οι μεταβάσεις, οι οποίες μπορούν να ανιχνευθούν από τον πράκτορά μας, αποτελούν την κατάσταση του περιβάλλοντος. Οι νέες καταστάσεις που παρατηρούνται από τον πράκτορά μας παράγουν ένα σήμα "ανταμοιβής". Συνδυάζοντας τις ενέργειες που πραγματοποιεί ο πράκτορας, τις μεταβάσεις μεταξύ καταστάσεων και την πιθανή ανταμοιβή σε κάθε μετάβαση, ο πράκτορας αρχίζει να θέτει τις βάσεις για ένα μοντέλο ενισχυτικής μάθησης. Υποθέτουμε ότι ο πράκτορας μας δεν έχει όλες τις πληροφορίες του περιβάλλοντος (π.χ. την ακριβή εναπομένουσα απόσταση από τον ιστό της σημαίας προορισμού) και κατέχει μόνο τη γνώση που εμφανίζεται στην οθόνη στην τρέχουσα κατάσταση (δηλ. το καρέ). Βασικές πληροφορίες που είναι διαθέσιμες σε μια κατάσταση είναι οι εξής:

- Θέση: οι συντεταγμένες του πράκτορα.
- Χρόνος: ο υπολειπόμενος χρόνος στο ρολόι
- Δράση: η στάση του πράκτορα
- Κατάσταση: η κατάσταση διαβίωσης και η λειτουργική κατάσταση ("μικρός", "ψηλός" και "πύρινη σφαίρα") του πράκτορα
- Σημαία: η κατάσταση της σημαίας στο τέλος του επίπεδου (ενεργοποιήστε στον τερματισμό).

Table 4.1: *Αναλυτικά οι πληροφορίες που μας επιστρέφει το περιβάλλον εκπαίδευσής με τη μέθοδο step*

Κλειδί	**Τύπος**	**Περιγραφή**
coins	int	Ο αριθμός των συλλεχθέντων κερμάτων
flag_get	bool	True αν ο Mario έφτασε σε μια σημαία ή ένα ax
life	int	Ο αριθμός των ζωών που απομένουν, δηλαδή 3, 2, 1
score	int	Το αθροιστικό σκορ στο παιχνίδι
stage	int	Το τρέχον στάδιο, δηλαδή 1, ..., 4
status	str	Η κατάσταση του Mario, δηλαδή 'small', 'tall', 'fireball'
time	int	Ο χρόνος που απομένει στο ρολόι
world	int	Ο τρέχων κόσμος, π.χ. 1, ..., 8
x_pos	int	Η θέση x του Mario στη σκηνή (από αριστερά)
y_pos	int	Η θέση y του Mario στη σκηνή (από κάτω)

Μεταξύ αυτών, οι πληροφορίες θέσης, χρόνου, δράσης χρησιμοποιούνται για την εξαγωγή της επόμενης κατάστασης και τον υπολογισμό της "ανταμοιβής" μιας δεδομένης κατάστασης, οι λεπτομέρειες της οποίας παρέχονται στην συνέχειά. Οι πληροφορίες κατάστασης και οι πληροφορίες για το κοντάρι της σημαίας χρησιμοποιούνται για να προσδιοριστεί αν το παιχνίδι έχει τελειώσει.

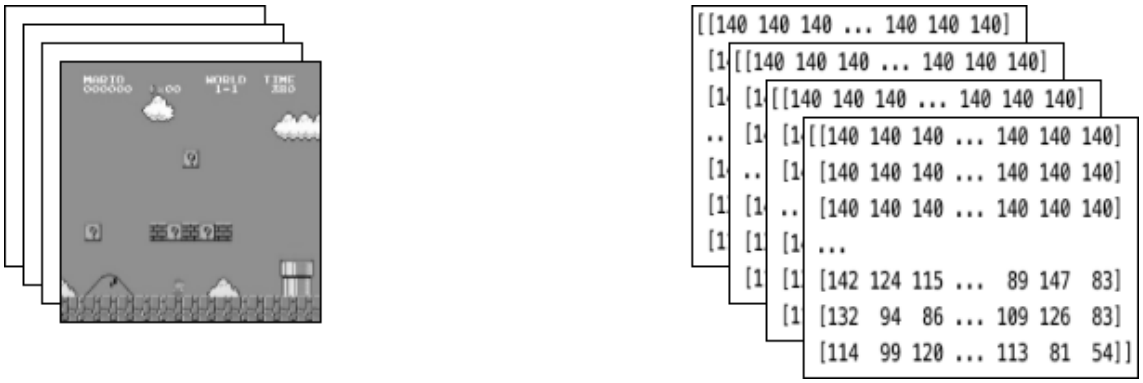
4.4 Επεξεργασία περιβάλλοντος

Ο αρχικός χώρος παρατήρησης για το Super Mario Bros είναι μια RGB εικόνα 240 x 256 x 3. Συχνά αυτή είναι περισσότερη πληροφορία από όση χρειάζεται ο πράκτοράς μας, για παράδειγμα, οι ενέργειες του Mario δεν εξαρτώνται από το χρώμα των σωλήνων ή του ουρανού. Προκειμένου να επιταχύνουμε τον χρόνο εκπαίδευσης του μοντέλου μας, χρησιμοποιήσαμε wrappers. Έτσι εφαρμόζουμε ορισμένους μετασχηματισμούς στα αρχικά δεδομένα του περιβάλλοντος πριν τα στείλουμε στον πράκτορα.

Μετατρέπουμε την εικόνα RGB σε κλίμακα του γκρι- με αυτόν τον τρόπο μειώνεται το μέγεθος της κατάστασης χωρίς να χάνονται χρήσιμες πληροφορίες. Τώρα το μέγεθος κάθε κατάστασης: [1, 240, 256] Με αυτό τον τρόπο το πρόβλημά μας γίνεται υπολογιστικά ευκολότερο, άρα μειώνετε ο χρόνος εκπαίδευσής.

Κάθε καρέ περικόπτεται σε μια τετραγωνική εικόνα NxN, δηλαδή κάθε κατάσταση στο περιβάλλον είναι ένα 4 x 84 x 84 x 84 x 1 (μια λίστα από 4 συνεχή καρέ 84 x 84 pixel γκρι κλίμακας), γεγονός που καθιστά ταχύτερη και ευκολότερη την επεξεργασία των λεπτομερειών μιας εικόνας από ένα νευρωνικό δίκτυο. Στοιβάζουμε πολλαπλά καρέ μεταξύ τους για να δείχνουν την κίνηση διαφόρων αντικειμένων. Αυτό βοηθά τον πράκτορα να καταλάβει, πώς να αντιδράσει σε μια συγκεκριμένη κατάσταση, καθώς είναι σε θέση να συμπεράνει πού θα βρίσκονται τα αντικείμενα σε μελλοντικές καταστάσεις. Αποτελεί παρόμοιο τρόπο με τον οποίο οι άνθρωποι ερμηνεύουν τα περιβάλλοντα και τις καταστάσεις. Αν παίρναμε ένα τυχαίο στιγμιότυπο οθόνης από ένα παιχνίδι και το μοιραζόμασταν με τον φίλο μας χωρίς κανένα πλαίσιο, μπορεί να δυσκολευόταν να προσδιορίσει προς ποια κατεύθυνση κινείται κάθε αντικείμενο στην οθόνη. Αν, ωστόσο, παίρναμε τέσσερα στιγμιότυπα οθόνης από τέσσερα συνδεδεμένα καρέ και μοιραζόμασταν αυτά τα στιγμιότυπα οθόνης με τη σειρά με τον φίλο

μας, πιθανότατα θα ήταν πιο εύκολο να αποκρυπτογραφήσει την κατεύθυνση της κίνησης για κάθε αντικείμενο στην εικόνα, καθώς θα μπορούσε να συγκρίνει τυχόν διαφορές μεταξύ της ακολουθίας. Για την μελέτη που ακολουθεί επιλέχτηκαν 4 καρέ.



Σχήμα 4.2: *Η τελική κατάσταση αποτελείται από 4 διαδοχικά καρέ με κλίμακα του γκρι που στοιβάζονται μεταξύ τους, όπως φαίνεται παραπάνω στην εικόνα στα αριστερά. Κάθε φορά που ο Mario πραγματοποιεί μια ενέργεια, το περιβάλλον ανταποκρίνεται με μια κατάσταση αυτής της δομής.*

Κεφάλαιο 5

Πειραματική Μελέτη

5.1 Συνάρτηση ανταμοιβής

Υποθέτουμε ότι ο στόχος του παιχνιδιού είναι να νικήσουμε το παιχνίδι όσο το δυνατόν γρηγορότερα. Για να το κάνουμε αυτό η συνάρτηση ανταμοιβής υποθέτει ότι ο στόχος του παιχνιδιού είναι να κινηθεί όσο το δυνατόν πιο δεξιά (να αυξήσει την τιμή x του πράκτορα), όσο το δυνατόν πιο γρήγορα, χωρίς να πεθάνει. Επομένως, οι πιο σημαντικές παράμετροι είναι η οριζόντια απόσταση Δx από το τρέχον σημείο προς τον τερματισμό και η χρονική διαφορά Δt μεταξύ του χρόνου έναρξης και του χρόνου λήξης. Εκτός από αυτά τα δύο, επίσης θεωρούμε ότι έχει σημασία αν ο Mario φτάνει τελικά στο τέλος του επίπεδου f.

Θέλουμε να ενθαρρύνουμε τις τροχιές που νικούν το παιχνίδι, αλλά αυτό δεν πρέπει να επηρεάζει υπερβολικά τη συνολική ανταμοιβή, επειδή κάποιες τροχιές που δεν καταφέρνουν να φτάσουν στον προορισμό μπορεί να είναι καλύτερες, αν κάποια λάθη εξαλειφθούν. Η ίδια λογική ισχύει και για μια ποινή θανάτου d, την οποία χρησιμοποιούμε για να προειδοποιήσουμε τον πράκτορα να αποφύγει το σημείο που πέθανε στο επόμενο επεισόδιο. Άρα για τη μοντελοποίηση του παιχνιδιού, χρησιμοποιούμε τέσσερις ξεχωριστές μεταβλητές που συνθέτουν την ανταμοιβή:

1. v: απόσταση, η διαφορά στις τιμές x του πράκτορα μεταξύ των καταστάσεων στην προκειμένη περίπτωση πρόκειται για τη στιγμιαία ταχύτητα για το συγκεκριμένο βήμα
   - $v = \Delta x_n = x_{n+1} - x_n$
     - x_n είναι η θέση x πριν από το βήμα
     - x_{n+1} είναι η θέση x μετά το βήμα
   - κίνηση προς τα δεξιά $\Leftrightarrow$ v > 0
   - κίνηση προς τα αριστερά $\Leftrightarrow$ v < 0
   - παραμένει στην ίδια θέση $\Leftrightarrow$ v = 0
2. c : η διαφορά στο ρολόι του παιχνιδιού μεταξύ των καρέ η ποινή εμποδίζει τον πράκτορα να παραμείνει ακίνητος.Ο χρόνος ξεκινά από 400 δευτερόλεπτα για κάθε επίπεδο και μειώνεται κάθε δευτερόλεπτο.
   - $c = \Delta t_n = t_n - t_{n+1}$

- t_n είναι η ένδειξη του ρολογιού πριν από το βήμα
- t_{n+1} είναι η ένδειξη του ρολογιού μετά το βήμα

- κανένα τικ του ρολογιού ⇔ c = 0
- χτύπημα ρολογιού ⇔ c < 0

3. d_n: μια ποινή θανάτου που τιμωρεί τον πράκτορα επειδή πεθαίνει σε μια κατάσταση. Η ποινή αυτή ενθαρρύνει τον πράκτορα να αποφύγει το θάνατο

   - ζωντανός ⇔ d = 0
   - νεκρός ⇔ d = σ, σ μια αρνητική σταθερά.

4. f_n: μια ανταμοιβή που επιβραβευθεί τον πράκτορα επειδή τερμάτισε το επίπεδο, αν συλλέξει τη σημαία (ή νικήσει τον Bowser) στο τέλος του επιπέδου για να τον ενθαρρύνει να νικήσει επιτυχώς το στάδιο.

   - τερματισμός ⇔ f = ρ , ρ μια θετική σταθερά.
   - αλλιώς ⇔ f = 0

Αν και αυτό είναι ένα αρκετά ισχυρό σύστημα ανταμοιβής, τα επίπεδα θα μπορούσαν να παιχτούν πιο "κανονικά" (όπως θα τα έπαιζαν οι περισσότεροι άνθρωποι), ανταμείβοντας τον Mario για την αύξηση του σκορ του στο παιχνίδι, νικώντας εχθρούς, μαζεύοντας νομίσματα και συλλέγοντας μανιτάρια. Αυτές οι πληροφορίες επιστρέφονται από το περιβάλλον εκπαίδευσής και στην συνάρτησή ανταμοιβής προστίθεται ένας ακόμα ορός s που επιβραβεύει τον πράκτορά όταν αυξάνετε το σκορ του παιχνιδιού. s = σκορ παιχνιδιού / 40

Έστω r_n η συνολική ανταμοιβή στο n-οστό βήμα ορίζετε ως το άθροισμά των παραπάνω μεταβλητών.

$$r_n = v + c + f_n + d_n + s$$

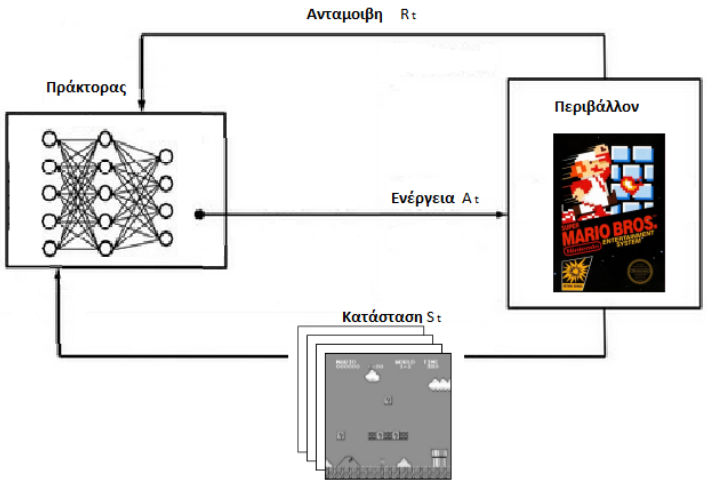
Τερματισμός επεισοδίων

Η εμπειρία στο παιχνίδι χωρίζεται σε επεισόδια. Στο τέλος κάθε επεισοδίου, το περιβάλλον επαναφέρεται στην αρχική του κατάσταση. Τα επεισόδια μπορούν να τερματιστούν κάτω από τις εξής συνθήκες:

- Ο παίκτης ολοκληρώνει ένα επίπεδο με επιτυχία. Σε αυτό το σημείο αναφοράς, η ολοκλήρωση ενός επιπέδου αντιστοιχεί στο πέρασμα μιας ορισμένης οριζόντιας μετατόπισης εντός του επιπέδου.
- Ο πράκτορας πεθαίνει από έναν εχθρό ή πέφτει σε μια τρυπά
- τελειώνει ο χρόνος του επίπεδου, αυτό ισοδυναμεί με 400 χρονικά βήματα.

5.1.1 Απεικόνιση του προβλήματος

Έχουμε αναφερθεί σε όλα τα στοιχεία που αποτελούν το πρόβλημά μας. Το πρόβλημα διατυπώνεται ως πρόβλημα απόφασης Markov. Όπου ο πράκτορας είναι οι αλγόριθμοι



Σχήμα 5.1

DQN, DDQN, A3C, PPO, που θα αλληλεπιδράσουν με το περιβάλλον. Το περιβάλλον, ο κόσμος στον οποίο λειτουργεί ο πράκτορας, είναι το ίδιο το περιβάλλον του παιχνιδιού, και οι καταστάσεις, πληροφορίες που έχει στη διάθεσή του ο πράκτορας σχετικά με το τρέχον περιβάλλον του, δίνονται από τον προσομοιωτή gym-super-mario-bros με τις τροποποιήσεις που αναφέραμε. Ενέργειες αποτελούν την αντίδραση που υιοθετεί ένας πράκτορας για να αλληλεπιδράσει με το περιβάλλον (εδώ, περπάτημα, τρέξιμο, άλμα κ.λπ.), αναλυτικά κεφάλαιο 4.2 Ανταμοιβή είναι η ανατροφοδότηση που λαμβάνει ο πράκτορας από το περιβάλλον για τις ενέργειές του, αναλυτικά κεφάλαιο 5.1.

5.1.2 Βιβλιοθήκες

Η υλοποίηση των αλγορίθμων σε αυτήν την εργασία πραγματοποιήθηκε στην γλώσσα προγραμματισμού Python. Η χρήση της σε εφαρμογές μηχανικής μάθησης είναι ευρεία διότι, το Tensorflow και το Pytorch, τα πιο γνωστά πακέτα μοντελοποίησης νευρωνικών δικτύων προσφέρουν εύχρηστα API σε Python. Μια από τις σπουδαίες πτυχές της Python είναι ότι υπάρχει μεγάλη πιθανότητα κάτι που χρειαζόμαστε να έχει ήδη δημιουργηθεί και να είναι διαθέσιμο σε μια βιβλιοθήκη. Μερικές από τις βασικές βιβλιοθήκες που χρησιμοποιήθηκαν στην παρούσα εργασία είναι οι ακόλουθες:

torch: Το Torch είναι μια βιβλιοθήκη μηχανικής μάθησης ανοικτού κώδικα, ένα πλαίσιο επιστημονικών υπολογισμών και μια γλώσσα σεναρίων βασισμένη στη γλώσσα προγραμματισμού Lua. Παρέχει ένα ευρύ φάσμα αλγορίθμων για βαθιά μάθηση

gym_super_mario_bros: όπως αναφέραμε και στο προηγούμενο κεφάλαιο,είναι ένα περιβάλλον συμβατό με το OpenAI Gym, το οποίο επιτρέπει στα προγράμματα Python να αλληλεπιδρούν απρόσκοπτα με το Super Mario Bros.

nes_py: Ένα πλαίσιο για τη διασύνδεση μεταξύ περιβαλλόντων παιχνιδιών NES, όπως το Super Mario Bros. και εφαρμογών Python.

rllib: Μια βιβλιοθήκη ενισχυτικής μάθησης βασισμένη πάνω στο Ray, η οποία περιλαμβάνει αρκετά ισχυρά μοντέλα και αλγορίθμους για την εύκολη δημιουργία εφαρμογών RL.

numpy: συντομογραφία του "Numerical Python", ειναι μια βιβλιοθήκη Python που χρησιμοποιείται για την εργασία με πίνακες. Διαθέτει επίσης συναρτήσεις για την εργασία στον τομέα της γραμμικής άλγεβρας, του μετασχηματισμού Fourier και των πινάκων.

matplotlib.pyplot: είναι μια συλλογή συναρτήσεων που κάνουν το matplotlib να λειτουργεί όπως το MATLAB. Κάθε συνάρτηση pyplot κάνει κάποια αλλαγή σε ένα σχήμα: π.χ. δημιουργεί ένα σχήμα, δημιουργεί μια περιοχή σχεδίασης σε ένα σχήμα, σχεδιάζει κάποιες γραμμές σε μια περιοχή σχεδίασης, διακοσμεί το γράφημα με ετικέτες, κ.λπ.

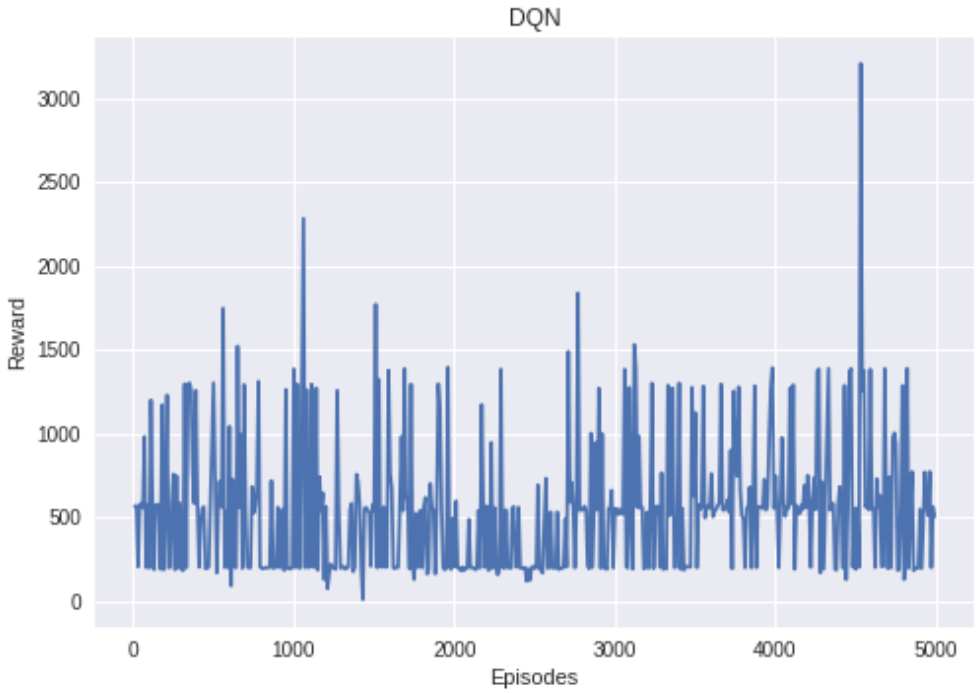
5.2 Παράμετροι Πειραμάτων και Αποτελέσματα

Τις υλοποιήσεις των αλγορίθμων εκτελέστηκαν σε υπολογιστή του εργαστηρίου Συστημάτων Τεχνητής Νοημοσύνης και Μάθησης με επεξεργαστή Intel(R) Core(TM) i7-5820K 3.30GHz και Κάρτα γραφικών GeForce GTX 1080 8 GB. Ο βελτιστοποιητής (optimizer) που χρησιμοποιήσαμε για τη βελτιστοποίηση της αντικειμενικής συνάρτησης στις υλοποιήσεις μας είναι ο Adam (adaptive moment estimation). Οι πράκτορες αξιολογήθηκαν ως προς την ανταμοιβή που συγκέντρωσαν σε κάθε επεισόδιο κατά την εκπαίδευσή τους και την ταχύτητα εκπαίδευσης ενώ παράλληλα απεικονίζουμε τον κυλιόμενο μέσο όρο των επεισοδιακών βαθμολογιών για όλα τα επεισόδια που ολοκληρώθηκαν μαζί με διάφορα άλλα στατιστικά στοιχεία, όπως τα ποσοστά ολοκλήρωσης των επιπέδων, τον βέλτιστο χρόνο ολοκλήρωσης και τον μέσο όρο ανταμοιβής των τελευταίων 100 επεισοδίων κάθε αλγορίθμου.

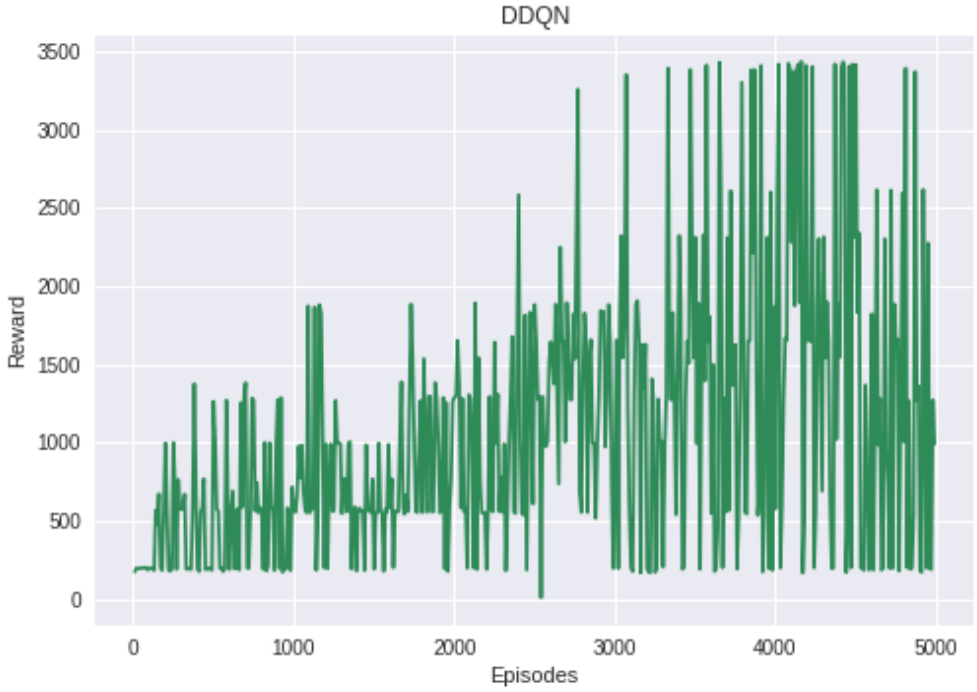
Table 5.1: *Παράμετροί DQN & DDQN*

Παράμετρος	**Τιμη**	**Περιγραφή**
world	1-1	Επιλογή επίπεδου
action-space	simple	Το σύνολο ενεργειών που μπορεί να χρησιμοποιήσει ο Mario όπως δίνεται από το gym-super-mario-bros.
lr	0.00025	Ο ρυθμός μάθησης που θα χρησιμοποιηθεί
max_memory_size	30000	
Batch size	32	
Dropout	0,2	
gamma	0.9	Καθορίζει τον συντελεστή προεξόφλησης που θα χρησιμοποιηθεί για τις ανταμοιβές
	0.99	Exploration decay
For epsilon greedy	1	Exploration_max,
	0.02	Exploration min*

*Στην αρχή της εξερεύνησης ο πράκτορας πραγματοποιεί τυχαίες δράσεις. Μετά από κάθε επεισόδιο, η βαθμός της εξερεύνησης θα φθίνει μέχρι να φτάσει σε ένα ελάχιστο 0,02.



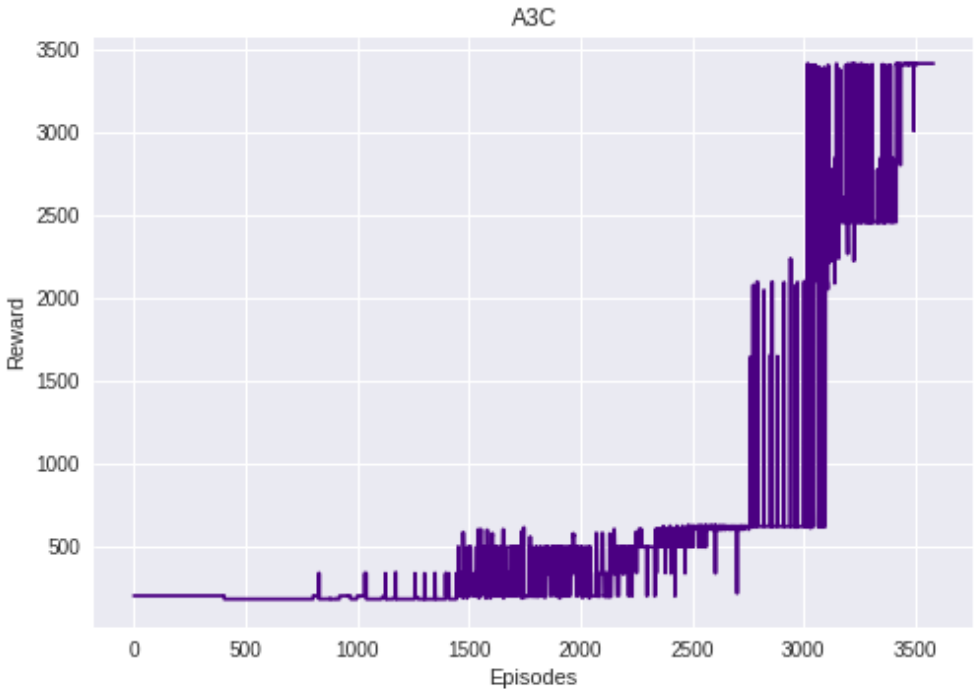
Σχήμα 5.2: *Καμπύλης εκμάθησης DQN*



Σχήμα 5.3: *Καμπύλης εκμάθησης DDQN*

Table 5.2: *Παράμετροι A3C*

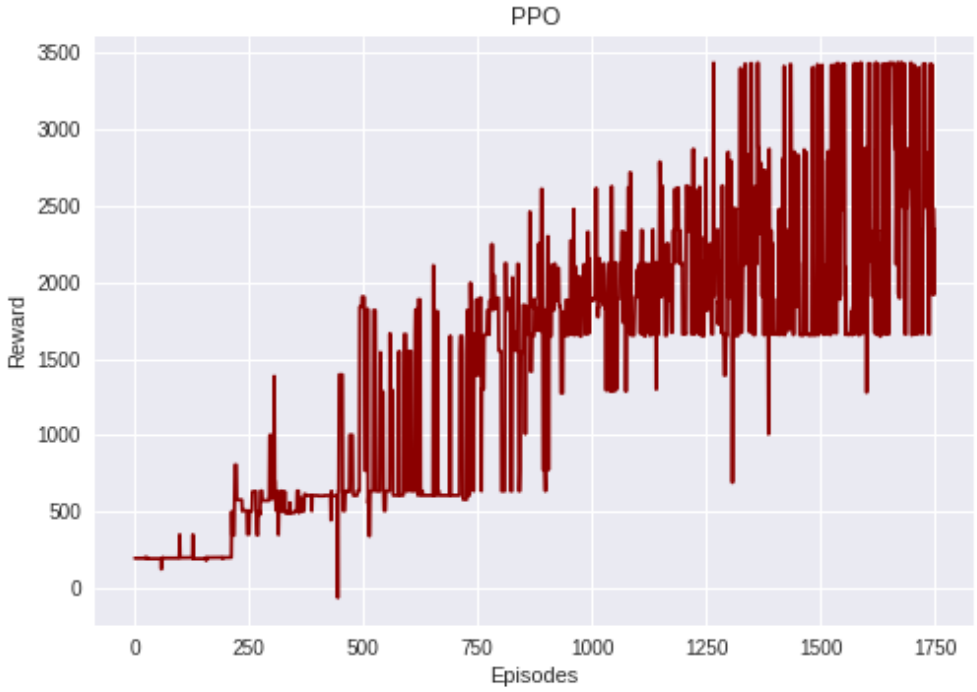
Παράμετρος	**Τιμή**	**Περιγραφή**
world	1-1	Επιλογή επίπεδου
action-space	Simple movement	Το σύνολο ενεργειών που μπορεί να χρησιμοποιήσει ο Mario όπως δίνεται από το gym-super-mario-bros.
beta	0.01	Ο συντελεστής που χρησιμοποιείται στον υπολογισμό της εντροπίας
gamma	0.9	Καθορίζει τον συντελεστή προεξόφλησης που θα χρησιμοποιηθεί για τις ανταμοιβές
learning-rate	1e-4	Ο ρυθμός μάθησης που θα χρησιμοποιηθεί.
max-actions	200	Ο αριθμός των ενεργειών που πρέπει να επαναληφθούν κατά τη διάρκεια της δοκιμής
num-episodes	3500	Ο αριθμός των επεισοδίων που θα εκτελεστούν στο συγκεκριμένο περιβάλλον
num-processes	4	Ο αριθμός των διεργασιών εκπαίδευσης που θα εκτελούνται παράλληλα.
tau	1.0	Η τιμή που χρησιμοποιείται για τον υπολογισμό του Generalized Advantage Estimator (GAE).



Σχήμα 5.4: *Καμπύλης εκμάθησης A3C*

Table 5.3: *Παράμετροι PPO*

Παράμετρος	**Τιμή**	**Περιγραφή**
world	1-1	Επιλογή επίπεδου
action-space	Simple movement	Το σύνολο ενεργειών που μπορεί να χρησιμοποιήσει ο Mario όπως δίνεται από το gym-super-mario-bros.
beta	0.01	Ο συντελεστής που χρησιμοποιείται στον υπολογισμό της εντροπίας
gamma	0.9	Καθορίζει τον συντελεστή προεξόφλησης που θα χρησιμοποιηθεί για τις ανταμοιβές
learning-rate	1e-4	Ο ρυθμός μάθησης που θα χρησιμοποιηθεί.
max-actions	200	Ο αριθμός των ενεργειών που πρέπει να επαναληφθούν κατά τη διάρκεια της δοκιμής
num-processes	4	Ο αριθμός των διεργασιών εκπαίδευσης που θα εκτελούνται παράλληλα.
tau	1.0	Η τιμή που χρησιμοποιείται για τον υπολογισμό του Generalized Advantage Estimator (GAE).
epsilon	0.2	Παράμετρος αποκοπής της αντικειμενικής



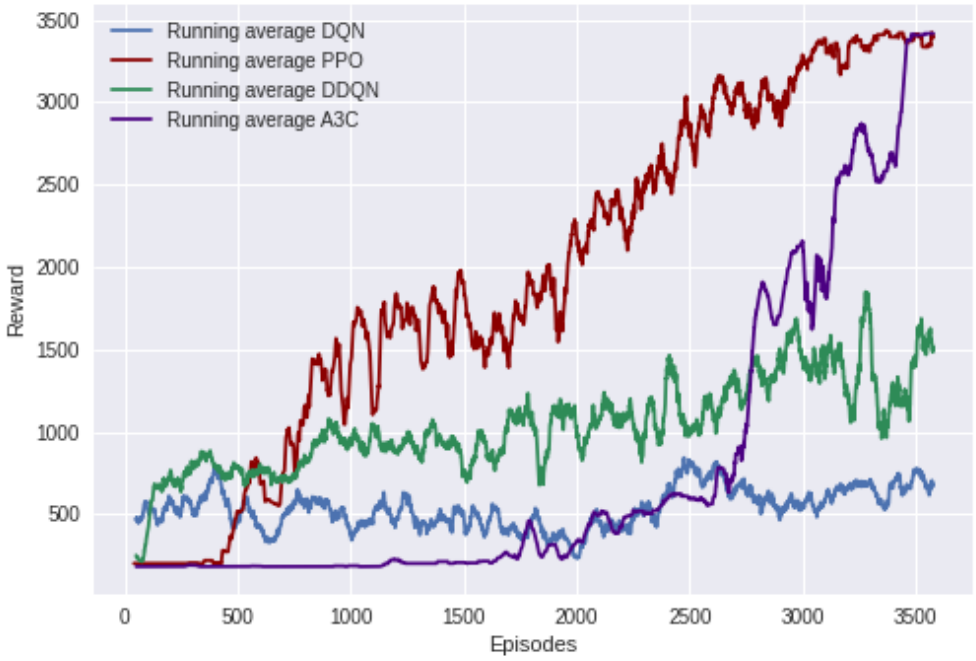
Σχήμα 5.5: *Καμπύλης εκμάθησης PPO*

Table 5.4: *Μέσος ορός ανταμοιβής των τελευταίων 100 επεισοδίων*

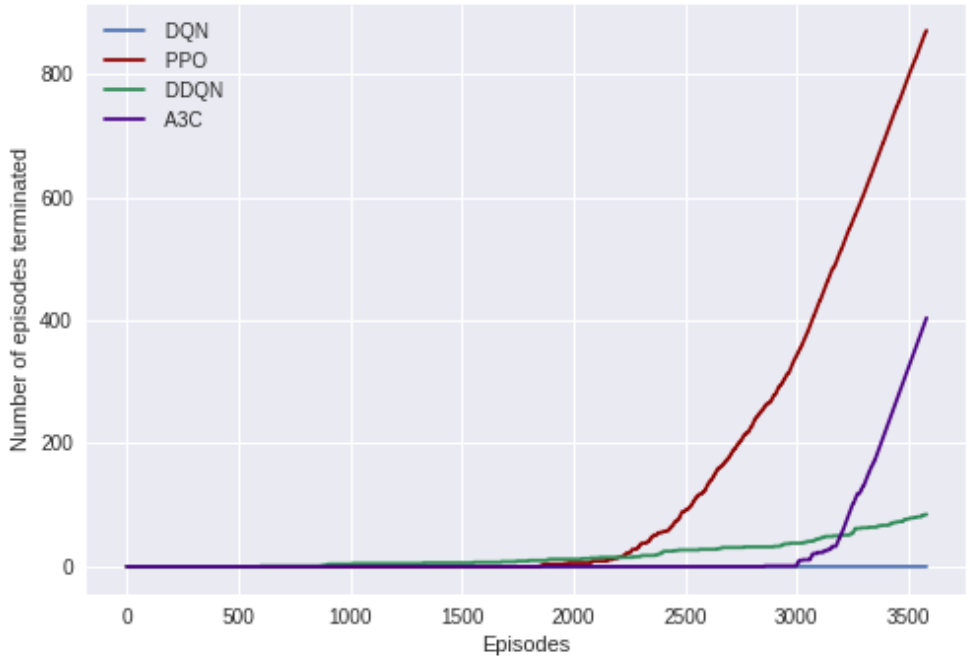
	DQN	**DDQN**	**A3C**	**PPO**
Ανταμοιβή	715.83	1542.48	3412.67	3385.905

Table 5.5: *Καλύτερος χρόνος τερματισμού*

	DQN	**Μέσος άνθρωπος**	**DDQN**	**A3C**	**PPO**	**Παγκόσμιο ρεκόρ**
Χρόνος	139	336	338	346	348	348



Σχήμα 5.6: *Καμπύλες κυλιόμενου μέσου όρου εκμάθησης των Αλγορίθμων με παράθυρο τα 50 επεισόδιά*



Σχήμα 5.7: *Αριθμός τερματισμού επίπεδων ανά επεισόδιο*

5.3 Παρατηρήσεις και Συμπεράσματα

Υπάρχει ένα όριο στη μεγίστη ανταμοιβή που που μπορούν να συλλέξουν οι πράκτορες στο 3500 καθώς τα επίπεδά του παιχνιδιού είναι πεπερασμένα. Το DQN χρησιμοποιεί την επανάληψη εμπειριών για να μαθαίνει από όλες τις προηγούμενες πολιτικές. Από μόνο του είναι ασταθές και δίνει κακή σύγκλιση, συνεπώς απαιτεί διάφορα πρόσθετα. Το DDQN αποτελεί βελτίωσή του DQN, χρησιμοποιεί δύο δίκτυα για να μειώσει την υπεραισιοδοξία του DQN, με αποτέλεσμα, ως ένα βαθμό, πιο σταθερή και αξιόπιστη μάθηση. Ωστόσο χρειάζονται πολλές επαναλήψεις ώστε να δώσει ικανοποιητικά αποτελέσματά.

Το A3C και το PPO δίνουν σημαντικά καλύτερα αποτελέσματα σε σύγκριση με τους DQN και Double DQN. Βασικό λόγο αποτελεί η ενσωμάτωση της μάθησης από διαφορετικές εμπειρίες με τη χρήση πολλαπλών επεξεργασιών. Μπορεί να παρατηρηθεί ότι το PPO παρέχει καλύτερο ρυθμό σύγκλισης και απόδοσης από τις άλλες τεχνικές, αλλά είναι ευαίσθητο στις αλλαγές. Παρόλο που παίρνει λιγότερο χρόνο για να εκπαιδευτεί, δίνει καλύτερα αποτελέσματα σε σύγκριση με άλλους αλγορίθμους Το A3C είναι ένας αλγόριθμος που συνδυάζει τόσο τη μάθηση βάσει αξίας όσο και τις κλίσεις πολιτικής. Ο πράκτορας εκτιμά τη δράση με βάση την πολιτική και ο κριτής εκτιμά τη συνάρτηση αξίας της δράσης που αναλαμβάνει ο δράστης και αξιολογεί την δράση. Δε δίνει καλύτερα αποτελέσματα στην αρχική φάση, ωστόσο έχει ραγδαία και σταθερή βελτίωσή μετά από έναν αριθμό επεισοδίων. Ο A3C επωφελείται όταν υπάρχει μεγάλη υπολογιστική ισχύς, και θα περιμέναμε αισθητή βελτίωσή αν τρέχαμε τον αλγόριθμο σε ένα ακόμα πιο δυνατό μηχάνημα. Η απόδοσή των αλγορίθμων ήταν η αναμενόμενη και συμφωνεί με τη βιβλιογραφία [38], [31], [34], [40], [41].

Θα πρέπει να σημειώσουμε ότι μέσω της κατασκευής της συνάρτησης ανταμοιβής επηρεάζουμε τη βέλτιστη πολιτική και θα περιμέναμε αλλαγές στην επίδοση των αλγορίθμων αλλάζοντας τα βάρη στην συνάρτησή ανταμοιβής ή κατασκευάζοντας μια νέα. [40] Επιπλέον οι υπερπαράμετροι θα μπορούσαν πάντα να χρήζουν βελτίωσης.

Συνοψίζοντας, οι μέθοδοι κλίσης πολιτικής (Policy Gradient) έχουν καλύτερη απόδοσή από τις μεθόδους Deep Q. Οι μέθοδοι Deep Q συχνά συγκλίνουν αλλά όχι στη βέλτιστη ανταμοιβή. Η PPO είχε την καλύτερη απόδοση. Οι μέθοδοι κλίσης πολιτικής εκπαιδεύονται πολύ ταχύτερα. Όλοι οι αλγόριθμοι είναι επαρκείς για την επίλυση του περιβάλλοντος.

5.4 Μελλοντικές κατευθύνσεις

Τα πειράματά χρειάζονται μεγάλο χρονικό διάστημα για να παράξουν αποτελέσματά και τα αποτελέσματά αυτά εξαρτώνται υπερβολικά από τις υπερπαραμέτρους. Κατά συνέπειά θα παρουσίαζε ενδιαφέρον η ανάπτυξη ενός αποδοτικού μηχανισμού ρύθμισης υπερπαραμέτρων καθώς τεχνικές όπως το grid-search και fine-tuning είναι ιδιαίτερα κοστοβόρες σε χρόνο υπολογισμού.

Επιπλέον θα μπορούσαμε να εξετάσουμε τη δυνατότητα για μαθησιακή μεταφορά (transfer learning), δηλαδή αν οι πράκτορες θα είναι σε θέση να χρησιμοποιήσουν την τρέχουσα εμπειρία τους σε αυτό το περιβάλλον, ώστε να επιλύουν περιβάλλοντα που δεν έχουν συναντήσει προηγουμένως. [42] Ένα τέτοια περιβάλλον θα μπορούσε να είναι το Super Mario Maker 2 όπου ο παίκτης μπορεί να κατασκευάσει επίπεδά χρησιμοποιώντας τα στοιχεία του

Super Mario Bros, δίνοντας την δυνατότητα σε άλλους παίχτες να τα παίξουν. Υπάρχουν έτσι μεγαλύτερα και δυσκολότερα επίπεδά από το Super Mario Bros. Οι τρέχουσες μέθοδοι εξακολουθούν να υστερούν σε δύο μέτωπα την αποτελεσματικότητα της μάθησης ως προς τα δεδομένα και όπως αναφέραμε την γενίκευση σε νέα περιβάλλοντα. Θα είχε μεγάλο ενδιαφέρον η διερεύνηση και εφαρμογή των πρόσφατων και εξαιρετικά υποσχόμενων ευρημάτων σχετικά με τη χρήση της επαύξησης δεδομένων στην RL [43] για την τεράστια αύξηση της αποτελεσματικότητας του δείγματος και της ισχύος της γενίκευσης.

Προκειμένου να ενθαρρυνθεί καλύτερα η εξερεύνηση του χώρου καταστάσεων, θα μπορούσε να χρησιμοποιηθεί μια μορφή εσωτερικού κινήτρου ανταμοιβής και όχι μια συνάρτηση ανταμοιβής κατασκευασμένη με το χέρι από εμάς, παρόμοια με το RND [44] και το Agent57 [45].

Ο τομέας της RL εξελίσσεται γρήγορα και νέοι αλγόριθμοι έχουν εμφανιστεί μετά τον PPO (που είναι ο νεότερος αλγόριθμος σε αυτό το σύνολο). Ωστόσο, οι αλγόριθμοι αιχμής δεν έχουν ακόμη υλοποιηθεί σε φιλικές προς τον χρήστη βιβλιοθήκες Python ανοικτού κώδικα και δεν έχουν συγκριθεί με καθιερωμένους αλγορίθμους. Ιστορικά, οι σύγχρονοι αλγόριθμοι RL καθίστανται όλο και περισσότερο εφαρμόσιμοι σε ένα ευρύτερο φάσμα ρυθμίσεων προβλημάτων και η απόδοσή τους αυξάνεται αναλογικά. Θα παρουσίαζε ιδιαίτερο ενδιαφέρον η μελέτη αλγόριθμων αιχμής, καθώς και σύγκριση με τις υπάρχουσες υλοποιήσεις ελπίζοντας ότι θα ξεπερνούν οποιονδήποτε από αυτές που παρουσιάζονται στην παρούσα εργασία.

Βιβλιογραφία

[1] Fabio De Sousa Ribeiro, Georgios Leontidis και Stefanos Kollias. *Capsule routing via variational bayes*. *Proceedings of the AAAI Conference on Artificial Intelligence*, τόμος 34, σελίδες 3749–3756, 2020.

[2] Fabio De Sousa Ribeiro, Georgios Leontidis και Stefanos Kollias. *Introducing routing uncertainty in capsule networks*. *Advances in Neural Information Processing Systems*, 33:6490–6502, 2020.

[3] Fabio De Sousa Ribeiro, Francesco Calivá, Mark Swainson, Kjartan Gudmundsson, Georgios Leontidis και Stefanos Kollias. *Deep bayesian self-training*. *Neural Computing and Applications*, 32(9):4275–4291, 2020.

[4] Dimitrios Kollias, Miao Yu, Athanasios Tagaris, Georgios Leontidis, Andreas Stafylopatis και Stefanos Kollias. *Adaptation and contextualization of deep neural network models*. *2017 IEEE symposium series on computational intelligence (SSCI)*, σελίδες 1–8. IEEE.

[5] Dimitrios Kollias και Stefanos Zafeiriou. *Training deep neural networks with different datasets in-the-wild: The emotion recognition paradigm*. *2018 International Joint Conference on Neural Networks (IJCNN)*, σελίδες 1–8. IEEE, 2018.

[6] Manolis Wallace, Nicolas Tsapatsoulis και Stefanos Kollias. *Intelligent initialization of resource allocating RBF networks*. *Neural Networks*, 18(2):117–122, 2005.

[7] Christopher M Bishop και Nasser M Nasrabadi. *Pattern recognition and machine learning*, τόμος 4. Springer, 2006.

[8] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.

[9] Shai Shalev-Shwartz και Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[10] Stuart J Russell και Peter Norvig. *Artificial intelligence: a modern approach. Malaysia*, 2016.

[11] Elizabeth G. E. Kyonka. *Law of Effect*, σελίδες 868–870. Springer US, Boston, MA, 2011.

[12] Alan M Turing. *Intelligent machinery, a heretical theory*. *The Turing test: Verbal behavior as the hallmark of intelligence*, 105, 1948.

[13] JH Andreae και BR Gaines. *An Introductory Description of the STeLLA Learning Machine*.

[14] Ronald A. Howard και James E. Matheson. *Risk-Sensitive Markov Decision Processes*. *Management Science*, 18(7):356–369, 1972.

[15] Richard S Sutton. *Learning to predict by the methods of temporal differences*. *Machine learning*, 3(1):9–44, 1988.

[16] A Harry Klopf. *A neuronal model of classical conditioning*. *Psychobiology*, 16(2):85–125, 1988.

[17] Yutian Chen, Aja Huang, Ziyu Wang, Ioannis Antonoglou, Julian Schrittwieser, David Silver και Nandode Freitas. *Bayesian Optimization in AlphaGo*. *CoRR*, abs/1812.06855, 2018.

[18] Ge Liu, Rui Wu, Heng-Tze Cheng, Jing Wang, Jayden Ooi, Lihong Li, Ang Li, Wai Lok Sibon Li, Craig Boutilier και Ed H. Chi. *Data Efficient Training for Reinforcement Learning with Adaptive Behavior Policy Sharing*. *CoRR*, abs/2002.05229, 2020.

[19] Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andrew Bolt και Charles Blundell. *Never Give Up: Learning Directed Exploration Strategies*. *CoRR*, abs/2002.06038, 2020.

[20] Yuxi Li. *Deep Reinforcement Learning*. *CoRR*, abs/1810.06339, 2018.

[21] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare και Joelle Pineau. *An Introduction to Deep Reinforcement Learning*. *CoRR*, abs/1811.12560, 2018.

[22] Richard S Sutton και Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[23] Csaba Szepesvári. *Algorithms for reinforcement learning*. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

[24] Warren B. Powell. *A unified framework for stochastic optimization*. *European Journal of Operational Research*, 275(3):795–821, 2019.

[25] S.S. Haykin. *Neural Networks and Learning Machines*. Pearson International Edition. Pearson, 2009.

[26] Leslie Pack Kaelbling, Michael L Littman και Andrew W Moore. *Reinforcement learning: A survey*. *Journal of artificial intelligence research*, 4:237–285, 1996.

[27] Geoffrey J Gordon. *Stable function approximation in dynamic programming*. *Machine learning proceedings 1995*, σελίδες 261–268. Elsevier, 1995.

[28] Martin Riedmiller. *Neural fitted Q iteration-first experiences with a data efficient neural reinforcement learning method*. *European conference on machine learning*, σελίδες 317–328. Springer, 2005.

[29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra και Martin A. Riedmiller. *Playing Atari with Deep Reinforcement Learning*. *CoRR*, abs/1312.5602, 2013.

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski και others. *Human-level control through deep reinforcement learning*. *nature*, 518(7540):529–533, 2015.

[31] Hadovan Hasselt, Arthur Guez και David Silver. *Deep Reinforcement Learning with Double Q-learning*. *CoRR*, abs/1509.06461, 2015.

[32] Yong Fang, Cheng Huang, Yijia Xu και Yang Li. *RLXSS: Optimizing XSS Detection Model to Defend Against Adversarial Attacks Based on Reinforcement Learning*. *Future Internet*, 11:177, 2019.

[33] Alekh Agarwal, Sham M. Kakade, Jason D. Lee και Gaurav Mahajan. *Optimality and Approximation with Policy Gradient Methods in Markov Decision Processes*. *CoRR*, abs/1908.00261, 2019.

[34] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver και Koray Kavukcuoglu. *Asynchronous Methods for Deep Reinforcement Learning*. *CoRR*, abs/1602.01783, 2016.

[35] Mohammad Babaeizadeh, Iuri Frosio, Stephen Tyree, Jason Clemons και Jan Kautz. *GA3C: GPU-based A3C for Deep Reinforcement Learning*. *CoRR*, abs/1611.06256, 2016.

[36] Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg και Koray Kavukcuoglu. *IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures*. *CoRR*, abs/1802.01561, 2018.

[37] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan και Pieter Abbeel. *Trust Region Policy Optimization*. *CoRR*, abs/1502.05477, 2015.

[38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford και Oleg Klimov. *Proximal Policy Optimization Algorithms*. *CoRR*, abs/1707.06347, 2017.

[39] Christian Kauten. *Super Mario Bros for OpenAI Gym*. GitHub, 2018.

[40] Thomas Nakken Larsen, Halvor Ødegård Teigen, Torkel Laache, Damiano Varagnolo και Adil Rasheed. *Comparing Deep Reinforcement Learning Algorithms' Ability to Safely Navigate Challenging Waters*. *Frontiers in Robotics and AI*, 8, 2021.

[41] Sudhir Yarram Velivela Vamsi Krishna. *COMPARISON OF REINFORCEMENT LEARNING ALGORITHMS*, 2020.

[42] Zhuangdi Zhu, Kaixiang Lin, Anil K. Jain και Jiayu Zhou. *Transfer Learning in Deep Reinforcement Learning: A Survey*, 2020.

[43] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel και Aravind Srinivas. *Reinforcement Learning with Augmented Data*, 2020.

[44] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell και Alexei A. Efros. *Large-Scale Study of Curiosity-Driven Learning*, 2018.

[45] Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andew Bolt και Charles Blundell. *Never Give Up: Learning Directed Exploration Strategies*, 2020.