ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Εφαρμογή Αλγορίθμων Ενισχυτικής Μάθησης και Μεταφορά Μάθησης στο Sonic the Hedgehog

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΧΡΥΣΟΣΤΟΜΟΣ Α. ΚΑΝΙΟΥΡΑΣ

Επιβλέπων Καθηγητής : Ανδρέας-Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2020

Εθνικο Μετσοβιο Πολυτεχνειο
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Εφαρμογή Αλγορίθμων Ενισχυτικής Μάθησης και Μεταφορά Μάθησης στο Sonic the Hedgehog

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΧΡΥΣΟΣΤΟΜΟΣ Α. ΚΑΝΙΟΥΡΑΣ

Επιβλέπων Καθηγητής : Ανδρέας-Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 10/7/2020.

.....	.....	.....
Ανδρέας - Γεώργιος Σταφυλοπάτης Καθηγητής Ε.Μ.Π	Γεώργιος Στάμου Καθηγητής Ε.Μ.Π.	Στέφανος Κόλλιας Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2020.

.....

Χρυσόστομος Α. Κανιούρας
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Τα τελευταία χρόνια, έχει γίνει εμφανές πως το πεδίο της Ενισχυτικής Μηχανικής Μάθησης μπορεί να επιλύσει προβλήματα υψηλών διαστάσεων δεδομένης καλής συνάρτησης κόστους και απεριόριστου χρόνου να διαδράσει με το περιβάλλον. Ωστόσο , αν και αυτή τα στοιχεία αποτελούν κλειδί στην εκμάθηση των πρακτόρων δεν είναι πρακτικά εφικτά ούτε μοναδικά. Ιδανικά ένας έξυπνος πράκτορας θα είναι σε θέση να γενικεύει μεταξύ καθηκόντων που του ανατίθενται και να χρησιμοποιεί προγενέστερες εμπειρίες προκειμένου να αποκτήσει νέες δυνατότητες πιο γρήγορα.

Στο πλαίσιο αυτής της εργασίας εξετάστηκε η απόδοση των πιο πρόσφατων αλγόριθμων ενισχυτικής μάθησης, Rainbow και PPO, στα video games της σειράς Sonic the Hedgehog και η δυνατότητά τους για μαθησιακή μεταφορά (transfer learning) σε αυτό το περιβάλλον. Δηλαδή να μάθουν να επιλύουν και να κερδίζουν σε περιβάλλοντα που δεν έχουν συναντήσει προηγουμένως εφόσον έχουν εκπαιδευτεί προηγουμένως σε παρόμοια.

Λέξεις Κλειδιά: Τεχνητή Νοημοσύνη, Ενισχυτική Μάθηση, Βαθιά Ενισχυτική Μάθηση, Νευρωνικά Δίκτυα, DQN, Rainbow, Policy Gradient, PPO, Transfer Learning, Video Games, Sonic Retro Contest

Abstract

In recent years, it has become clear that the field of Reinforced Machine Learning can solve problems of high dimensions given a good cost function and unlimited time to interact with the environment.
However, while these elements are key to learning agents, they aren't practical or unique. Ideally, a smart agent will be able to generalize between assigned tasks and use previous experiences in order to gain new knowledge faster.
In this paper, we examined the performance of some of the most recent reinforcement learning algorithms, Rainbow and PPO, in video games of the series Sonic the Hedgehog as well as their potential for transfer learning in this environment. That is, to learn to solve and win in environments that have not been encountered before if they have been previously trained in similar ones.

Keywords: Artificial Intelligence, Reinforcement Learning, Deep Reinforcement Learning, Neural Networks, DQN, Rainbow, Policy Gradient, PPO, Transfer Learning, Video Games, Sonic Retro Contest

Περιεχόμενα

Κατάλογος Πινάκων

1 Εισαγωγή

Η ιδέα πως μαθαίνουμε αλληλεπιδρώντας με το περιβάλλον μας είναι ίσως η πρώτη που μας έρχεται κατα νου όταν αναλογιζόμαστε τη φύση της μάθησης. Όταν ένα παιδί παίζει, ψάχνει με τα χέρια του,ή κοιτάζει γύρω του, δεν έχει ρητό δάσκαλο, αλλά έχει μια άμεση αισθητήρια σύνδεση με το περιβάλλον του. Η άσκηση αυτής της σύνδεσης παράγει πληθώρα πληροφοριών για την αιτία και το αποτέλεσμα, για τις συνέπειες των ενεργειών και για το πώς πρέπει να ενεργήσει για την επίτευξη των στόχων. Καθ ΄όλη τη διάρκεια της ζωής μας, τέτοιες αλληλεπιδράσεις αποτελούν αναμφίβολα μια σημαντική πηγή γνώσης για το περιβάλλον και τον εαυτό μας. Είτε μαθαίνουμε να οδηγούμε ένα αυτοκίνητο ή να κάνουμε μια συνομιλία, έχουμε πλήρη επίγνωση του τρόπου με τον οποίο το περιβάλλον μας ανταποκρίνεται σε αυτό που κάνουμε και επιδιώκουμε να επηρεάσουμε τι συμβαίνει μέσω της συμπεριφοράς μας. Η μάθηση βασισμένη στην αλληλεπίδραση είναι μια θεμελιώδης ιδέα που βασίζεται σχεδόν σε όλες τις θεωρίες μάθησης και νοημοσύνης.

Η ενίσχυτική μάθηση εξερευνά ιδανικές καταστάσεις προς μάθηση και αξιολογεί την αποτελεσματικότητα των διαφόρων μεθόδων μάθησης. Δηλαδή υιοθετούμε την προοπτική ενός ερευνητή ή μηχανικού τεχνητής νοημοσύνης. Εξετάζουμε σχέδια για μηχανές που είναι αποτελεσματικές στην επίλυση μαθησιακών προβλημάτων επιστημονικού ή οικονομικού ενδιαφέροντος, αξιολογώντας σχέδια μέσω μαθηματικής ανάλυσης ή υπολογιστικών πειραμάτων. Αυτή η προσέγγιση που ερευνούμε είναι πολύ πιο επικεντρωμένη στην εκμάθηση που κατευθύνεται με άξονα κάποιον στόχο και με μέσο την αλληλεπίδραση, σε σύγκριση με άλλες προσεγγίσεις μηχανικής μάθησης.

2 Θεωρητικό υπόβαθρο

2.1 Μηχανική μάθηση

2.1.1 Ορισμός

Η μηχανική μάθηση (Machine Learning) είναι η επιστημονική μελέτη αλγορίθμων και στατιστικών μοντέλων που χρησιμοποιούν τα ηλεκτρονικά συστήματα για να εκτελέσουν μια συγκεκριμένη εργασία βασιζόμενα σε μοτίβα και συμπεράσματα, χωρίς να χρησιμοποιούν συγκεκριμένες οδηγίες και κανόνες. Θεωρείται υποσύνολο της τεχνητής νοημοσύνης. Οι αλγόριθμοι μηχανικής μάθησης δημιουργούν ένα μαθηματικό μοντέλο που βασίζεται σε δεδομένα δείγματος, γνωστό ως ¨δεδομένα εκπαίδευσης¨, προκειμένου να προβεί σε προβλέψεις ή αποφάσεις χωρίς να έχει προγραμματιστεί ρητά για την εκτέλεση της διαδικασίας.

2.1.2 Χρησιμότητα Μηχανικής μάθησης

Υπάρχουν αρκετοί λόγοι για τους οποίους η μηχανική μάθηση είναι σημαντική ειδικά για την περίπτωση πρακτόρων που ενεργούν και εξερευνούν κάποιο περιβάλλον:

- Τέτοιου είδους προβλήματα δεν μπορούν να μοντελοποιηθούν αποδοτικά έως καθόλου, παρά μόνο με παραδείγματα. Δηλαδή, σε ορισμένες περιπτώσεις είναι εύκολο να προσδιοριστούν ζεύγη εισόδου/εξόδου, αλλά όχι ένας συνοπτικός τρόπος συσχέτισης αυτών. Σε αυτές τις περιπτώσεις αλγόριθμοι μηχανικής μάθησης είναι ικανοί να προσαρμόσουν την εσωτερική τους δομή, ώστε να παράξουν σωστά αποτελέσματα για μεγάλο αριθμό δεδομένων.

- Είναι πιθανό, ανάμεσα σε μεγάλο όγκο δεδομένων να υπάρχει συσχέτιση, την οποία αλγόριθμοι μηχανικής μάθησης να μπορούν εύκολα να διακρίνουν. (Εξόρυξη δεδομένων)

- Εργαλεία μηχανικής μάθησης, είναι εξαιρετικά χρήσιμα, όταν το περιβάλλον στο οποίο θα χρησιμοποιηθούν έχει χαρακτηριστικά και παραμέτρους άγνωστες στο σχεδιαστή. Σε αυτή την περίπτωση, αλγόριθμοι μηχανικής μάθησης ανταποκρίνονται καλύτερα από κλασικές μεθόδους επίλυσης του προβλήματος.

- Τα περιβάλλοντα συνήθως είναι χρονικά μεταβαλλόμενα. Σε αυτή την περίπτωση υπολογιστικά συστήματα μηχανικής μάθησης που προσαρμόζονται στα δεδομένα εισόδου, προσφέρουν μια καλή λύση, σε αντίθεση με προγραμματισμένα εργαλεία, που υστερούν ως προς την προσαρμοστικότητα.

2.1.3 Μέθοδοι Μηχανικής μάθησης

Οι τύποι των αλγορίθμων μηχανικής μάθησης διαφέρουν ως προς την προσέγγισή τους, τον τύπο των δεδομένων που εισάγουν και εξάγουν και τον τύπο

της εργασίας ή του προβλήματος που προορίζονται για επίλυση. Το επιστημονικό πεδίο της , έχει κατηγοριοποιηθεί αντιστοίχως σε αρκετούς κλάδους διαφορετικών τύπων. Η πιο διαδεδομένη διάκριση αφορά την συσχέτιση δεδομένων εισόδου-εξόδου και τον τρόπο εκμετάλλευσης των σημάτων εξόδου, κατά τη διάρκεια της εκπαίδευσης.

Επιβλεπόμενη Μάθηση

Οι αλγόριθμοι επιβλεπόμενης μάθησης δημιουργούν ένα μαθηματικό μοντέλο ενός συνόλου δεδομένων που περιέχει τόσο τις εισόδους όσο και τις επιθυμητές εξόδους. Τα δεδομένα είναι γνωστά ως δεδομένα εκπαίδευσης και αποτελούνται από ένα σύνολο παραδειγμάτων. Κάθε παράδειγμα εκπαίδευσης έχει μία ή περισσότερες εισόδους και την επιθυμητή έξοδο, γνωστή και ως εποπτικό σήμα. Η μαθηματική προτυποποίηση κάθε τέτοιου παραδείγματος αντιπροσωπεύεται από μια συστοιχία ή διάνυσμα, που ονομάζεται διάνυσμα χαρακτηριστικών και τα δεδομένα εκπαίδευσης αντιπροσωπεύονται από ένα πίνακα. Μέσω επαναληπτικής βελτιστοποίησης μιας αντικειμενικής συνάρτησης κόστους, οι αλγόριθμοι επιβλεπόμενης μάθησης μαθαίνουν μια συνάρτηση που μπορεί να χρησιμοποιηθεί για την πρόβλεψη της εξόδου που σχετίζεται με νέες εισόδους που δεν έχει συναντήσει προηγουμένως.

Μη Επιβλεπόμενη Μάθηση

Οι μη επιβλεπόμενοι αλγόριθμοι μάθησης λαμβάνουν ένα σύνολο δεδομένων που περιέχει μόνο εισόδους και αναζητούν δομή στα δεδομένα, όπως ομαδοποίηση σημείων δεδομένων. Οι αλγόριθμοι, επομένως, μαθαίνουν από δεδομένα δοκιμών που δεν έχουν επισημανθεί, ταξινομηθεί ή κατηγοριοποιηθεί, σε αντίθεση με την επιβλεπόμενη. Αντί να ανταποκρίνονται σε ανατροφοδότηση, οι αλγόριθμοι μάθησης χωρίς επίβλεψη αναγνωρίζουν συνήθη χαρακτηριστικά των δεδομένων και αντιδρούν με βάση την παρουσία ή την απουσία τέτοιων κοινών στοιχείων σε κάθε νέο κομμάτι δεδομένων. Μια κεντρική εφαρμογή της μάθησης χωρίς επίβλεψη είναι στον τομέα της εκτίμησης πυκνότητας πθανότητας , στατιστικών και άλλες μεθόδους που συνοψίζουν και εξηγούν χαρακτηριστικά δεδομένων.

Η ανάλυση συμπλέγματος (cluster analysis) είναι η ανάθεση ενός συνόλου παρατηρήσεων σε υποσύνολα (ονομάζονται συμπλέγματα) έτσι ώστε οι παρατηρήσεις μέσα στο ίδιο σύμπλεγμα να είναι παρόμοιες σύμφωνα με ένα ή περισσότερα προκαθορισμένα κριτήρια, . Διαφορετικές τεχνικές ομαδοποίησης κάνουν διαφορετικές υποθέσεις για τη δομή των δεδομένων, που συχνά καθορίζονται από κάποια μετρική ομοιότητας και αξιολογούνται, για παράδειγμα, από την εσωτερική συμπάγεια ή την ομοιότητα μεταξύ των μελών της ίδιας ομάδας και τον διαχωρισμό, δηλαδή τη διαφορά μεταξύ συστάδων. Άλλες μέθοδοι βασίζονται στην εκτιμώμενη πυκνότητα και συνδεσιμότητα γραφήματος.

Ενισχυτική Μάθηση

Η Ενισχυτική Μάθηση είναι ένας τομέας μηχανικής μάθησης που ασχολείται με τον τρόπο με τον οποίο πράκτορες λογισμικού πρέπει να αναλάβουν δράση

σε ένα περιβάλλον, ώστε να μεγιστοποιήσουν κάποια έννοια σωρευτικής ανταμοιβής. Λόγω της γενικότητάς του, ο τομέας μελετάται σε πολλούς άλλους κλάδους, όπως η θεωρία παιγνίων, η θεωρία ελέγχου, η βελτιστοποίηση βάσει προσομοίωσης, τα συστήματα πολλαπλών πρακτόρων, η ευφυΐα των σμήνων, οι στατιστικές, οι γενετικοί αλγόριθμοι και άλλα. Στη ενισχυτική μάθηση, το περιβάλλον τυπικά μονετελοποιείται ως διαδικασία λήψης αποφάσεων Markov (Markov Decision Process), επειδή πολλοί αλγόριθμοι ενισχυτικής μάθησης χρησιμοποιούν τεχνικές δυναμικού προγραμματισμού. Αυτοί οι αλγόριθμοι δεν λαμβάνουν γνώση ενός ακριβούς μαθηματικού μοντέλου του MDP. Αντιθέτως χρησιμοποιούνται όταν ακριβή μοντέλα δεν είναι εφικτά. Εμφανίζονται σε αυτόνομα οχήματα ή στην εκμάθηση ενός πράκτορα να παίζει ένα παιχνίδι εναντίον ενός ανθρώπινου αντιπάλου. Αυτός ο κλάδος της Μηχανικής Μάθησης αποτελεί το πλαίσιο στο οποίο μοντελοποιούμε και αναλύουμε το πρόβλημα αυτής της εργασίας.

2.2 Τεχνητά Νευρωνικά Δίκτυα

Τα τεχνητά νευρωνικά δίκτυα (Artificial Neural Networks,ANN) είναι υπολογιστικά μοντέλα εμπνευσμένα από τα βιολογικά νευρωνικά δίκτυα του εγκέφαλο των ζώων. Όπως περιγράφηκε προηγουμένως για το γενικότερο πλαίσιο της μηχανικής μάθησης, τα συστήματα αυτά μαθαίνουν να εκτελούν εργασίες εξετάζοντας παραδείγματα, χωρίς να προγραμματίζονται με συγκεκριμένους κανόνες εργασίας. Η μοντελοποίηση νευρώνων ξεκίνησε περίποου την δεκαετία του 1940 από τους McCulloch και Pits και άνθησε στις επόμενες, κυρίως την δεκαετία του 1970 όταν επινοήθηκαν ο αυτόματος διαφορισμός (automatic differentiation) και η διάδοση προς τα πίσω (back propagation), η χρήση των οποίων έκανε εφικτή την εκπαίδευση των νευρώνων. Ωστόσο η εκτενής χρήση τους και κατέπέκταση η άνθηση του πεδίου της βαθιάς μηχανικής μάθησης , που βασίζεται κατέξοχήν σε αυτά, συνέβη τις τελευταίες δύο δεκαετίες, όταν η υπολογιστική δύναμη των μονάδων επεξεργασίας γραφικών (GPU) και των κατανεμημένων συστημάτων επέτρεψαν την χρήση μεγάλων τέτοιων δικτύων. Τέτοια μοντέλα γρήγορα διέπρεψαν στο πεδίο της αναγνώρισης εικόνας και όρασης υπολογιστών.

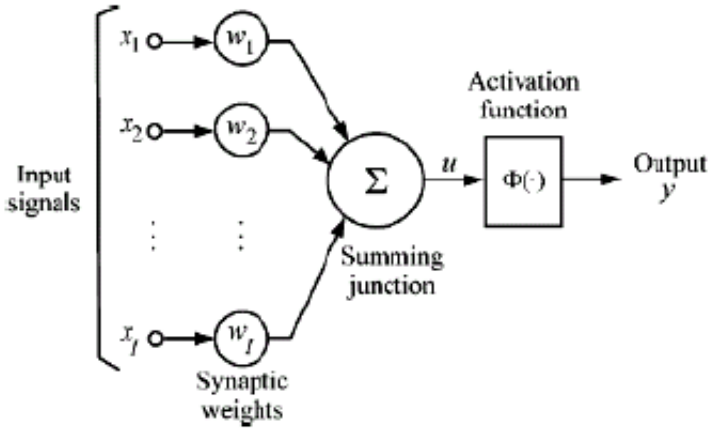
Αυτές οι μέθοδοι χρησιμοποιούν μια σειρά αρχιτεκτονικών νευρωνικών δικτύων, συμπεριλαμβανομένων συνελικτικών δικτύων, πολυστρωματικών perceptrons, περιορισμένων μηχανών Boltzmann, επαναλαμβανόμενων νευρωνικών δικτύων κ.α., και εκμεταλλεύονται τόσο την επιβλεπόμενη όσο και την μη επιβλεπόμενη διδασκαλία. Ήταν φυσικό να αναρωτηθούμε αν παρόμοιες τεχνικές θα μπορούσαν επίσης να ωφελήσουν την ενισχυτική μάθηση.

2.2.1 Τεχνητός Νευρώνας

Τα Νευρωνικά Δίκτυα αποτελούνται από τεχνητούς νευρώνες που διατηρούν κάποιες βιολογικές λειτουργίες των νευρώνων, δηλαδή λαμβάνουν πληροφορίες ως πολλαπλές εισόδους, τις συνδυάζουν με την εσωτερική τους κατάσταση (ενεργοποίηση) και ένα προαιρετικό κατώφλι χρησιμοποιώντας μια συνάρτηση ενεργοποίησης και παράγουν έξοδο χρησιμοποιώντας μια συνάρτηση εξόδου. Οι αρχικές είσοδοι είναι εξωτερικά δεδομένα, όπως εικόνες. Οι τελικές έξοδοι ολοκληρώνουν την εργασία, όπως η αναγνώριση ενός αντικειμένου σε μια εικόνα. Το σημαντικό χαρακτηριστικό της συνάρτησης ενεργοποίησης είναι ότι παράγει μια ομαλή, διαφορίσιμη αλλαγή στην έξοδο καθώς αλλάζει η τιμή εισόδου, δηλ. μια μικρή αλλαγή στην είσοδο παράγει μια μικρή αλλαγή στην έξοδο.

Στο Σχήμα 1 φαίνεται ένας νευρώνας. Απαρτίζεται από έναν γραμμικό συνδυασμό των εισόδων του, ο οποίος στην συνέχεια περνά από μια συνάρτηση (συνήθως μη γραμμική, αλλά διαφορίσιμη σχεδόν παντού). Αναλυτικότερα, ο αθροιστής (Σ) υπολογίζει τον γραμμικό συνδυασμό της εισόδου x_i με τα βάρη w_i του νευρώνα και προσθέτει και έναν σταθερό όρο b (το κατώφλι).

$$u = \sum_{i=1}^{m} x_i w_i + b \tag{1}$$



Σχήμα 1: Σχηματική αναπαράσταση ενός νευρώνα

Εν συνεχεία, το αποτέλεσμα του αθροιστή (u) περνά ως είσοδος στην συνάρτηση ενεργοποίησης $\varphi()$.

2.2.2 Συνάρτηση Ενεργοποίησης

Η συνάρτηση ενεργοποίησης αποφασίζει εάν ένας νευρώνας πρέπει να ενεργοποιηθεί ή όχι, εισάγοντας μια μη γραμμικότητα στην έξοδο. Μια χαρακτηριστική συνάρτηση ενεργοποίησης είναι η $\phi(u) = sgn(u)$, δηλαδή $+1$ όταν η είσοδος είναι θετική και με -1 όταν η είσοδος είναι αρνητική. Ο νευρώνας αυτός αναφέρεται ως perceptron. Ο στόχος του είναι να ταξινομεί ορθά, ομάδες δεδομένων $[x_1, x_2, ..., x_m]$ σε δύο διαφορετικές κλάσεις, άνω και κάτω του υπερεπιπέδου $\sum_{i=1}^{m} x_i w_i + b = 0$. Άλλες κλασικές συναρτήσεις που χρησιμοποιήθηκαν δίνονται παρακάτω, η καθεμία με τη δική της έννοια της απόστασης του σημείου από το υπερεπίπεδο.

Σιγμοειδής συνάρτηση:

$$S(x) = \frac{1}{1 - e^{-x}} \tag{2}$$

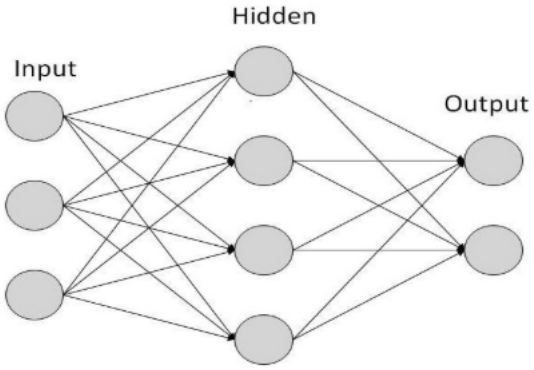
Rectified Linear Unit (ReLU):

$$f(x) = x^{+} = max(0, x) \tag{3}$$

2.2.3 Νευρωνικά Δίκτυα

Ένα νευρωνικό δίκτυο είναι ένας κατευθυνόμενος γράφος, όπου κάθε κόμβος είναι ένας νευρώνας συνδεδεμένος με άλλους μέσω συνδέσμων που αντιστοιχούν στις βιολογικές συνάψεις. Μία βασική αρχιτεκτονική ν.δ. είναι αυτή του εμπρόσθιου (Feed Forward Neural Network).

Το συγκεκριμένο νευρωνικό δίκτυο είναι χωρισμένο σε επίπεδα, όπου το κάθε επίπεδο αποτελείται απο έναν ορισμένο αριθμό νευρώνων. Η ροή επεξεργασίας της πληροφορίας, γίνεται σειριακά ανά επίπεδο από την αρχή προς το τέλος. Αυτή η δομή φαίνεται αναλυτικά και στο σχήμα 2.



Σχήμα 2: Σχηματική αναπαράσταση ενός Δικτύου

Κάθε νευρωνικό δίκτυο, εμπεριέχει τις εξής κατηγορίες επιπέδων:
Επίπεδο Εισόδου. Αυτό το επίπεδο, δέχεται τα δεδομένα εισόδου. Παρέχει δηλαδή πληροφορία από το περιβάλλον στο δίκτυο, χωρίς περαιτέρω επεξεργασία. Ο νευρώνες αυτοί, δηλαδή,νμεταβιβάζουν απλώς την πληροφορία στο κρυφό επίπεδο.
Κρυφό Επίπεδο. Μπορεί να είναι παραπάνω από ένα επίπεδο, τα οποία επεξεργάζονται τα δεδομένα εισόδου και εξάγουν τα κατάλληλα χαρακτηριστικά, τα οποία μεταβιβάζουν στο επίπεδο εξόδου. Καθώς αυξάνεται το βάθος των κρυφών επιπέδων τόσο αυξάνεται και το βάθος των χαρακτηριστικών που εξάγονται.
Επίπεδο Εξόδου. Με την επεξεργασία των δεδομένων από τα ενδιάμεσα επίπεδα,σε αυτό το επίπεδο λαμβάνεται μία απόφαση από το δίκτυο.

2.2.4 Μέθοδοι κλίσης (gradient descent and ascent)

Για να μιλήσουμε για το πώς μαθαίνει ένα νευρωνικό δίκτυο, πρέπει πρώτα να διατυπώσουμε τη βάση μας για τις μεθόδους βελτιστοποίησης που χρησιμοποιούνται, δηλαδή οι μέθοδοι κλίσης (gradient methods). Στη βελτιστοποίηση, η μέθοδος κλίσης είναι ένας αλγόριθμος για την επίλυση προβλημάτων της μορφής:

$$\min_{x \in \mathbb{R}^n} f(x)$$

με τη κατεύθύνση αναζήτησης να καθορίζεται από την κλίση της συνάρτησης στο τρέχον σημείο. Ένα παράδειγμα μεθόδου κλίσης, και ένα σημαντικό σημείο της θεωρίας των νευρωνικών δικτύων είναι ο αλγόριθμος σύγκλισης με ελάττωση της παραγώγου ή αλλιώς κατάβαση κλίσης (gradient descent), ένας επαναληπτικός αλγόριθμος βελτιστοποίησης πρώτης τάξης για την εύρεση τοπικού ελάχιστου μιας διαφοροποιήσιμης συνάρτησης.

Για να βρούμε ένα τοπικό ελάχιστο μιας συνάρτησης χρησιμοποιώντας (gradient descent), παίρνουμε βήματα ανάλογα με το αρνητικό της κλίσης (ή κατά προσέγγιση κλίση) της συνάρτησης στο τρέχον σημείο. Αλλά αν πάρουμε αντ άυτού βήματα ανάλογα με το θετικό της κλίσης, πλησιάζουμε ένα τοπικό μέγιστο αυτής της λειτουργίας. η διαδικασία είναι τότε γνωστή ως ανάβαση κλίσης (gradient ascent).

Η κατάβαση κλίσης βασίζεται στην παρατήρηση πως εάν η συνάρτηση πολλών μεταβλητών $F(\mathbf{x})$ ορίζεται και είναι παραγωγίσιμη σε γειτονιά ενός σημείου $\mathbf{a}$ τότε η $F(\mathbf{x})$ μειώνεται με το μεγαλύτερο ρυθμό εάν κινηθούμε από το $\mathbf{a}$ στην κατεύθινση της αρνητικής κλίσης της F στο $\mathbf{a}$, δηλαδή: $-\nabla F(\mathbf{a})$. Ακολοθεί πως αν:

$$\mathbf{a}_{n+1} = \mathbf{a}_n - \gamma \nabla F(\mathbf{a}_n)$$

για αρκετά μικρό $\gamma \in \mathbb{R}^+$, τότε $F(\mathbf{a}_n) \geq F(\mathbf{a}_{n+1})$.

Με άλλα λόγια, ο όρος $\gamma \nabla F(\mathbf{a}_n)$ αφαιρείται από το $\mathbf{a}$ επειδή θέλουμε να κινηθούμε ενάντια στην κλίση, προς το τοπικό ελάχιστο. Με αυτή την παρατήρηση, ξεκινάμε με τυχαίο x_0 για ένα τοπικό ελάχιστο του F και θεωρούμε την ακολουθία $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, ..$ τέτοια ώστε:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma \nabla F(\mathbf{x}_n)$$

Τότε έχουμε μια μονοτονική ακολουθία : $F(\mathbf{x}_0) \geq F(\mathbf{x}_1) \geq F(\mathbf{x}_2) \geq ...$ και ελπίζουμε ότι η ακολουθία $(\mathbf{x}_n)$ θα συγκλίνει σε επιθυμητό τοπικό ελάχιστο.

Με ορισμένες υποθέσεις σχετικά με τη συνάρτηση F (για παράδειγμα, F κυρτή και ∇F Lipschitz) και για συγκεκριμένες τιμές του γ εγγυόμαστε σύγκλιση προς ένα τοπικό ελάχιστο.

2.2.5 Οπίσθια Διάδοση

Η εκμάθηση είναι η διαδικασία προσαρμογής του δικτύου για να χειριστεί καλύτερα ένα έργο, λαμβάνοντας υπόψη τις παρατηρήσεις του δείγματος. Περιλαμβάνει την προσαρμογή των βαρών του δικτύου για τη βελτίωση της ακρίβειας του αποτελέσματος. Αυτό γίνεται με την ελαχιστοποίηση των παρατηρούμενων σφαλμάτων ως προς την *συνάρτηση απώλειας, κόστους ή σφάλματος* μέσω του αλγορίθμου της *οπίσθιας διάδοσης*.

Μια συνάρτηση απώλειας ή μια συνάρτηση κόστους είναι μια συνάρτηση που αντιστοιχεί ένα συμβάν ή τιμές μιας ή περισσοτέρων μεταβλητών σε έναν πραγματικό αριθμό $\mathcal{L}(\cdot)$ και διαισθητικά αντιπροσωπεύει κάποιο 'κόστος' που σχετίζεται με το συμβάν. Ένα πρόβλημα βελτιστοποίησης επιδιώκει να ελαχιστοποιήσει μια συνάρτηση απώλειας. Στην περίπτωσή μας συγκεκριμένα η απώλεια είναι κάποια έννοια διαφοράς ή απόστασης μεταξύ της επιθυμητής (y) και της πραγματικής ($\hat{y}$) εξόδου του νευρωνικού δικτύου $\mathcal{L}(\hat{y}, y)$.

Η συνάρτηση απώλειας, πρέπει να είναι κάτω φραγμένη, με το ελάχιστο να συμβαίνει όταν η πρόβλεψη είναι σωστή. Οι παράμετροι, συνεπώς, του δικτύου μετά την εκπαίδευση, θα έχουν τις κατάλληλες τιμές ώστε η $\mathcal{L}$ να έχει ελαχιστοποιηθεί.

Θεωρώντας μια συνάρτηση εισόδου - εξόδου $y = f(x; \theta)$ την οποία προσπαθεί να προσεγγίσει το δίκτυο, προκύπτει ότι $\mathcal{L}(\hat{y}, y) = \mathcal{L}(\hat{y}, f(x; \theta)) = \mathcal{L}(\theta)$, όπου ϑ τα βάρη του νευρωνικού δικτύου. Συνεπώς, στόχος της εκπαίδευσης είναι η επιλογή των βαρών θ_d , τέτοια ώστε να επιτευχθεί η σχέση: $\theta_d = \arg\min_\theta \mathcal{L}(\theta)$

Η οπίσθια διάδοση (backpropagation ,backprop, BP) είναι ένας αλγόριθμος που χρησιμοποιείται ευρέως στην εκπαίδευση των feedforward νευρωνικών δικτύων. Γενικεύσεις της υπάρχουν για άλλα τεχνητά νευρωνικά δίκτυα και για συναρτήσεις γενικότερα - ως "back-propagation" αναφέρεται μια γενική κατηγορία αλγορίθμων. Υπολογίζει την παράγωγο (gradient) της συνάρτησης απώλειας ως προς τα βάρη του δικτύου για ένα μόνο παράδειγμα εισόδου-εξόδου και το κάνει αποτελεσματικά σύμφωνα με τον κανόνα της αλυσίδας. Υπολογίζει τις παραγώγους ένα στρώμα τη φορά, επαναλαμβάνοντας τη διαδικασία προς τα πίσω με βάση το τελευταίο στρώμα αποφεύγοντας τους περιττούς επανυπολογισμούς των ενδιάμεσων όρων στον κανόνα της αλυσίδας. Αυτή η απόδοση καθιστά εφικτή τη χρήση των μεθόδων κλίσης που αναφέραμε για την εκπαίδευση δικτύων πολλαπλών στρωμάτων, μέσω της ενημέρωσης των βαρών για την ελαχιστοποίηση της απώλειας

Η λειτουργία του είναι ως εξής:

- Υπολογισμός της εξόδου για δεδομένη είσοδο $\hat{y} = f(x)$ και του αντίστοιχου σφάλματος σύμφωνα με τη συνάρτηση απώλειας $E = \mathcal{L}(\hat{y}, y)$
- Υπολογισμός της παραγώγου της απώλειας E ως προς κάθε βάρος w_{ij} : $\frac{\partial E}{\partial w_{ij}}$
- Αλλαγή της τιμής του βάρους κατά : $\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$, όπου η είναι η παράμετρος 'ρυθμός εκμάθησης' και συζητάται αργότερα.

Μπορούμε να εξετάσουμε το παραπάνω περισσότερο: για κάθε νευρώνα j, ορίζουμε την έξοδο o_j=$\phi(net_j)$, όπου $net_j = u_j = \sum_{i=1}^{m} x_i w_{ij} + b$. Η μερική

παράγωγος του σφάλματος μπορεί να γραφεί τώρα:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}} \tag{4}$$

Ωστόσο $\frac{\partial net_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}}(\sum_{k=a}^{b} o_k w_{kj} + b) = \frac{\partial}{\partial w_{ij}}(w_{ij}o_i) = o_i$.
Η παράγωγος $\frac{\partial o_j}{\partial net_j}$ θεωρείται γνωστή σύμφωνα με την συνάρτηση ενεργοποίησης κατά την κατασκευή του ν.δ. Τέλος η παράγωγος $\frac{\partial E}{\partial o_j} = \frac{\partial E(net_u, net_v, .., net_w)}{\partial o_j}$ όπου $L = \{u, v, .., w\}$ το σύνολο των νευρώνων που έχουν το o_j ως είσοδο. Η ολική παράγωγος ως προς o_j γίνεται τώρα

$$\frac{\partial E}{\partial o_j} = \sum_{l \in L}\left(\frac{\partial E}{\partial o_l}\frac{\partial o_l}{\partial net_l}w_{jl}\right) \tag{5}$$

Φαίνεται τώρα η αναδρομική φύση του αλγορίθμου και ο λόγος που χρησιμοποιείται δυναμικός προγραμματισμός. Συνδυάζοντας τις (4),(5):

$$\frac{\partial E}{\partial w_{ij}} = o_i \delta_j \tag{6}$$

με

$$\delta_j = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial net_j} = \begin{cases} \left(\sum_{l \in L} w_{jl}\delta_l\right)\frac{d\phi(net_j)}{dnet_j}, & \text{αν j είναι εσωτερικός νευρώνας} \\ \frac{\partial E}{\partial o_j}\frac{d\phi(net_j)}{dnet_j}, & \text{αν j είναι νευρώνας εξόδου} \end{cases}$$

όπου L το σύνολο των νευρώνων που δέχονται τον νευρώνα j ως είσοδο, (δηλ. τα δ_l είναι προυπολογισμένα) και η ενημέρωση των βαρρών γίνεται: $\Delta w_{ij} = -\eta o_i \delta_j$, με την αναδρομική σχέση των δ_j προφανή.

2.3 Ενισχυτική Μάθηση

2.3.1 Εισαγωγή

Η μάθηση μέσω αλληλεπίδρασης είναι θεμελιώδης ιδέα που υπογραμμίζει σχεδόν όλες τις θεωρίες μάθησης και νοημοσύνης. Είτε μαθαίνουμε να οδηγούμε ένα αμάξι ή να πραγματοποιήσουμε μια συνομιλία, έχουμε πλήρη επίγνωση του τρόπου με τον οποίο το περιβάλλον μας ανταποκρίνεται σε αυτό που κάνουμε και επιδιώκουμε να επηρεάσουμε τι συμβαίνει μέσω της συμπεριφοράς μας. Σε αυτό το κεφάλαιο μελετάται μία υπολογιστική προσέγγιση στην μάθηση μέσω αλληλεπίδρασης. Εξερευνούμε εξιδανικευμένες μαθησιακές καταστάσεις και αξιολογούμε την αποτελεσματικότητα των διαφόρων μεθόδων μάθησης

Σκοπός της ενισχυτικής μάθησης είναι ο πράκτορας να μάθει πώς να πράττει - πώς να αντιστοιχεί τις καταστάσεις (states) στις δράσεις (actions) - ώστε να μεγιστοποιήσει ένα αριθμητικό σήμα ανταμοιβής. Ο μαθητευόμενος δεν ενημερώνεται άμεσα (όπως στην επιβλεπόμενη μάθηση) για τις ενέργειες που πρέπει να αναλάβει, αλλά πρέπει να ανακαλύψει ποιες ενέργειες αποδίδουν τη μεγαλύτερη ανταμοιβή με διαρκείς , επαναλαμβανόμενες δοκιμές. Στις πιο ενδιαφέρουσες και προκλητικές περιπτώσεις, οι ενέργειες μπορεί να επηρεάσουν όχι μόνο την άμεση ανταμοιβή αλλά και την επόμενη κατάσταση και, μέσω αυτής, όλες τις επόμενες ανταμοιβές. Αυτά τα δύο χαρακτηριστικά -η αναζήτηση μέσω δοκιμών και σφαλμάτων και η αναβαλόμενη ανταμοιβή- είναι τα δύο πιο σημαντικά διακριτικά χαρακτηριστικά της ενισχυτικής μάθησης.

Επισημοποιούμε το πρόβλημα της ενισχυτικής μάθησης χρησιμοποιώντας ιδέες από τη θεωρία των δυναμικών συστημάτων, συγκεκριμένα, τον βέλτιστο έλεγχο των μη-γνωστών διαδικασιών απόφασης Markov (incompletely-known Markov decision processes). Η βασική ιδέα είναι απλά να συλλάβουμε τις πιο σημαντικές πτυχές του πραγματικού προβλήματος που αντιμετωπίζει ένας πράκτορας μάθησης που αλληλεπιδρά με το περιβάλλον ,με την πάροδο του χρόνου προκειμένου να επιτευχθεί ένας στόχος. Ένας πράκτορας μάθησης πρέπει να είναι σε θέση να αντιλαμβάνεται την κατάσταση του περιβάλλοντός του σε κάποιο βαθμό και πρέπει να μπορεί να λάβει αποφάσεις και δράσεις που επηρεάζουν την μετάβαση σε επόμενη κατάσταση. Ο πράκτορας πρέπει επίσης να έχει στόχο ή στόχους σχετικά με την κατάσταση του περιβάλλοντος. Οι διαδικασίες απόφασης Markov προορίζονται να περιλαμβάνουν μόνο αυτές τις τρεις πτυχές- αντίληψη, δράση και στόχο-στις απλούστερες δυνατές μορφές τους χωρίς να υποβαθμίζουν κανένα από αυτά. Οποιαδήποτε μέθοδος είναι κατάλληλη για την επίλυση τέτοιων προβλημάτων, θεωρούμε ότι είναι μια μέθοδος ενισχυτικής μάθησης.

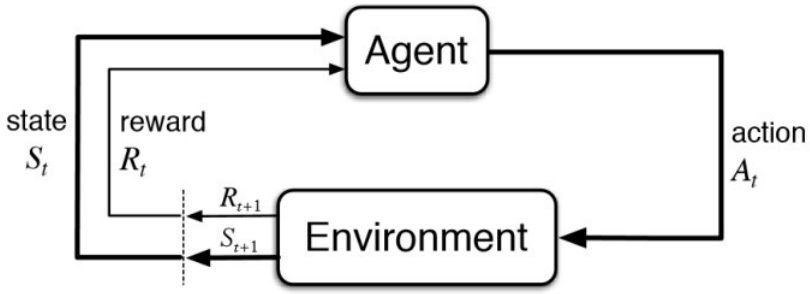
2.3.2 Μοντελοποίηση

Εισάγουμε το τυπικό πρόβλημα των πεπερασμένων διαδικασιών απόφασης Markov, ή των πεπερασμένων MDPs το οποίο περιλαμβάνει την διαρκή επαναξιολόγηση καταστάσεων (μέσω ανατροφοδοτήσεων), και την επιλογή διαφορετικών ενεργειών σε διαφορετικές καταστάσεις. Τα MDPs είναι η κλασσική μοντελο-

ποίηση της διαδοχικής λήψης αποφάσεων, όπου οι δράσεις επηρεάζουν όχι μόνο άμεσες ανταμοιβές, αλλά και μεταγενέστερες καταστάσεις και ,μέσω αυτών, μελλοντικών ανταμοιβών. Επομένως, οι MDPs περιλαμβάνουν την καθυστερημένη (ή αναβαλόμενη) ανταμοιβή και την αντιστάθμιση της άμεσης και καθυστερημένης ανταμοιβής.

Οι MDPs είναι μια μαθηματικά εξιδανικευμένη μορφή του προβλήματος της ενισχυτικής μάθησης για την οποία μπορούν να γίνουν ακριβείς θεωρητικές δηλώσεις. Εισάγουμε βασικά στοιχεία της μαθηματικής δομής του προβλήματος όπως οι επιστροφές (returns), οι συναρτήσεις αξίας (value functions) και οι εξισώσεις Bellman.

Ειδικότερα, ο πράκτορας και το περιβάλλον αλληλεπιδρούν σε κάθε ένα από μια σειρά διακριτών χρονικών βημάτων, $t = 0, 1, 2, 3, ...$ Σε κάθε βήμα t, ο πράκτορας λαμβάνει κάποια αναπαράσταση της κατάστασης του περιβάλλοντος, $S_t \in \mathcal{S}$, και με βάση αυτή επιλέγει μια δράση, $a_t \in \mathcal{A}$. Στην γενικότερη περίπτωση το πεδίο των δράσεων (action space) μπορεί να αλλάζει με το χρόνο (δηλαδή $\mathcal{A} = \mathcal{A}(s)$). Στη συγκεκριμένη εφαρμογή και για την απλούστευση του συμβολισμού το αφήνουμε $\mathcal{A}$.



Σχήμα 3: Σχηματική αναπαράσταση ενός Πράκτορα

Μετά το πέρας της χρονικής στιγμής t και λόγω της επίδρασης a_t , το περιβάλλον αλλάζει κατάσταση S_{t+1} και ο πράκτορας λαμβάνει αμοιβή (ή κέρδος) $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$. Τελικά ο πράκτορας μαζί με την MDP δημιουργούν μία σειρά από τούπλες κατάστασης, δράσης , κέρδους $(S_0, A_0, R_1, S_1, A_1, R_2, ...)$ που ονομάζουμε τροχιά. Στην περίπτωση των πεπερασμένων αλυσίδων (και στην εφαρμογή μας) τα σύνολα $\mathcal{S}, \mathcal{A}, \mathcal{R}$ είναι όλα πεπερασμένα. Σε αυτήν την περίπτωση οι μεταβλητές R_t, S_t έχουν καλώς ορισμένες κατανομές πιθανότητας και βασίζονται στην προϊστορία της αλυσίδας. Δηλαδή για συγκεκριμένες τιμές των τυχαίων μεταβλητών , $s' \in \mathcal{S}$ και $r \in \mathcal{R}$, και δεδομένης της προηγούμενης κατάστασης και δράσης, υπάρχει συγκεκριμένη πιθανότητα αυτές οι τιμές να εμφανιστούν την χρονική στιγμή t:

$$p(s', r|s, a) \doteq Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$$

Να σημειώσουμε πως η κατάσταση στην οποία βασίζεται η μετάβαση μπορεί να είναι (και συνήθως είναι) διαφορετική από την τελευταία παρατήρηση. Στην

γενικότερη περίπτωση μπορεί να είναι το σύνολο όλων των προηγούμενων παρατηρήσεων.

Οι πιθανότητες που δίδονται από τη συνάρτηση τεσσάρων ορισμάτων p χαρακτηρίζουν εντελώς τη δυναμική μιας πεπερασμένης MDP. Από αυτήν, μπορεί κανείς να υπολογίσει οτιδήποτε άλλο επιθυμεί να μάθει για το περιβάλλον, όπως οι πιθανότητες μετάβασης από μία φάση s σε άλλη s' δεδομένης δράσης a.

$$p(s'|s,a) = \sum_{r \in \mathcal{R}} p(s', r|s, a)$$

Επίσης τα περιβάλλοντα, ειδικά στις προσομοιώσεις, μπορούν να είναι ντετερμινιστικά τόσο ως προς την μετάβαση της κατάστασης από s σε s' όσο και ως προς το επιστρεφόμενο κέρδος r. Η παραπάνω εξίσωση παραμένει ως έχει.

Σκοπός και Αμοιβές

Ο σκοπός του πράκτορα εκφράζεται,όπως έχουμε αναφέρει, όσον αφορά την αμοιβή (ή κέρδος). Είναι ένα σήμα που παράγεται από το περιβάλλον στον πράκτορα ως απόκριση στις δράσεις του. Είναι απλώς ένας πραγματικός αριθμός που παράγεται σε κάθε χρονική στιγμή $R_t \in \mathbb{R}$. Σκοπός του πράκτορα είναι η μακροπρόθεσμη μεγιστοποίηση του συνολικού κέρδους. Αυτή η ιδέα εκφράζεται επίσημα ως η *υπόθεση κέρδους*:

> *Οποιοδήποτε σκοπός και στόχος μπορεί να μοντελοποιηθεί ως η μεγιστοποίηση της αναμενόμενης τιμής της συσώρευσης ενός βαθμωτού σήματος που ονομάζουμε αμοιβή(ή κέρδος).*

Για έναν πράκτορα που μαθαίνει να παίζει το παιχίδι που μας ενδιαφέρει (και θα περιγράψουμε αργότερα αναλυτικά), οι προφανείς ανταμοιβές είναι ένας θετικός αριθμός για νίκη, αρνητικός για ήττα και 0 για τις υπόλοιπες περιπτώσεις. Πιο απλά για έναν πράκτορα που μαθαίνει να παίζει σκάκι, θα ήταν $+1$ για νίκη, -1 για ήττα και 0 για ισοπολία και για όλες τις μη τερματικές θέσεις.

Είναι καθοριστικής σημασίας οι ανταμοιβές που δημιουργούμε να δείχνουν πραγματικά τι θέλουμε να επιτύχουμε. Συγκεκριμένα, το σήμα ανταμοιβής δεν είναι η πληροφορία που μεταδίδει στον πράκτορα προηγούμενη γνώση για το πώς να επιτύχουμε αυτό που θέλουμε να κάνει. Για παράδειγμα, στο σκάκι ο πράκτορας πρέπει να ανταμείβεται μόνο για την πραγματική νίκη, όχι για την επίτευξη δευτερευόντων στόχων (που πιθανώς να βοηθούν) όπως η λήψη των κομματιών του αντιπάλου του ή να αποκτήσει τον έλεγχο του κέντρου του ταμπλό. Εάν η επίτευξη αυτών των ειδών δευτερευόντων στόχων ανταμείβεται, τότε ο πράκτορας μπορεί να βρει έναν τρόπο να τα επιτύχει χωρίς να επιτύχει τον πραγματικό κύριο στόχο. Για παράδειγμα, θα μπορούσε να βρει έναν τρόπο να πάρει τα κομμάτια του αντιπάλου, θυσιάζοντας το ίδιο το παιχνίδι.Το σήμα ανταμοιβής είναι ο τρόπος που επικοινωνούμε στο ρομπότ αυτό που θέλουμε να επιτύχει, όχι πώς θέλουμε να επιτευχθεί.

Επεισόδια και Επιστροφή

Σε γενικές γραμμές, επιδιώκουμε να μεγιστοποιήσουμε την αναμενόμενη επιστροφή (ή απόδοση), όπου η επιστροφή, που δηλώνεται ως G_t, ορίζεται ως κάποια συγκεκριμένη συνάρτηση της ακολουθίας ανταμοιβής. Στην απλούστερη περίπτωση η απόδοση είναι το άθροισμα των ανταμοιβών:

$$G_t \doteq R_{t+1} + R_{t+2} + ... + R_T$$

όπου T είναι ένα τελικό χρονικό βήμα. Αυτή η προσέγγιση έχει νόημα σε εφαρμογές στις οποίες υπάρχει μια φυσική έννοια του τελευταίου χρονικού βήματος, δηλαδή, όταν η αλληλεπίδραση μεταξύ πράκτορα-περιβάλλοντος διαρείται με προφανή τρόπο σε υποακολουθίες που ονομάζουμε επεισόδια, όπως οι παρτίδες ενός παιχνιδιού, τα ταξίδια μέσα σε ένα λαβύρινθο ή κάθε είδους επαναλαμβανόμενη αλληλεπίδραση. Κάθε επεισόδιο τελειώνει σε μια ειδική κατάσταση που ονομάζεται τερματική, ακολουθούμενη από μια επαναφορά σε μια τυπική εναρκτήρια κατάσταση ή σε ένα δείγμα από μια τυπική κατανομή των καταστάσεων εκκίνησης. Ακόμα κι αν φανταστούμε τα επεισόδια να τελειώνουν με διαφορετικούς τρόπους, όπως η νίκη και η απώλεια ενός παιχνιδιού, το επόμενο επεισόδιο αρχίζει ανεξάρτητα από πώς τελείωσε το προηγούμενο. Επομένως όλα τα επεισόδια μπορούν να θεωρηθούν ότι τελειώνουν στην ίδια τερματική κατάσταση, με διαφορετικές ανταμοιβές για τα διαφορετικά αποτελέσματα. Οι εργασίες με επεισόδια αυτού του είδους ονομάζονται επεισοδιακές εργασίες. Ο χρόνος τερματισμού, T, είναι μια τυχαία μεταβλητή που κανονικά ποικίλλει από επεισόδιο σε επεισόδιο.

Από την άλλη πλευρά, σε πολλές περιπτώσεις η αλληλεπίδραση μεταξύ πράκτορα και περιβάλλοντος δεν διαιρείται φυσικά σε αναγνωρίσιμα επεισόδια, αλλά συνεχίζει χωρίς περιορισμό. Για παράδειγμα, αυτός θα είναι ο φυσικός τρόπος για να διατυπωθεί ένα συνεχές πρόβλημα ελέγχου, ή μια εφαρμογή σε ένα ρομπότ με μεγάλη διάρκεια ζωής. Ονομάζουμε αυτές τις διεργασίες συνεχόμενες. Η κατασκευή της επιστροφής γίνεται προβληματική για συνεχόμενα καθήκοντα επειδή το τελικό χρονικό βήμα γίνεται $T = \infty$ και η επιστροφή, που προσπαθούμε να μεγιστοποιήσουμε, θα μπορούσε εύκολα να είναι άπειρη. (Για παράδειγμα, ας υποθέσουμε ότι ο πράκτορας λαμβάνει μια ανταμοιβή $+1$ σε κάθε βήμα του χρόνου). Χρησιμοποιούμε συνήθως έναν ορισμό της επιστροφής που είναι ελαφρώς πιο πολύπλοκος εννοιολογικά αλλά πολύ απλούστερος μαθηματικά.

Η πρόσθετη έννοια που χρειαζόμαστε είναι αυτή της εκπτωτικής επιστροφής (discounting returns). Σύμφωνα με αυτή την προσέγγιση, ο πράκτορας προσπαθεί να επιλέξει δράσεις έτσι ώστε να μεγιστοποιηθεί το άθροισμα των μειωμένων ανταμοιβών που λαμβάνει στο μέλλον. Συγκεκριμένα, επιλέγει το A_t για να μεγιστοποιήσει την αναμενόμενη μειωμένη απόδοση:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

όπου γ είναι μια παράμετρος, $0 \leq \gamma \leq 1$, που ονομάζεται ρυθμός έκπτωσης (discount rate).

Ο ρυθμός γ καθορίζει τη τρέχουσα αξία των μελλοντικών ανταμοιβών: μια ανταμοιβή που εισπράχθηκε k χρονικά βήματα στο μέλλον αξίζει μόνο γ^{k-1} φορές

απ΄ ότι θα άξιζε αν είχε ληφθεί αμέσως. Αν $\gamma < 1$, το άπειρο άθροισμα έχει μια πεπερασμένη τιμή εφόσον η ακολουθία ανταμοιβής $\{R_k\}$ είναι φραγμένη. Αν $\gamma = 0$, ο πράκτορας είναι ΄μυωπικός΄ και ασχολείται μόνο με τη μεγιστοποίηση των άμεσων ανταμοιβών: ο στόχος του σε αυτή την περίπτωση είναι να μάθει πώς να επιλέξει το A_t έτσι ώστε να μεγιστοποιηθεί μόνο το R_{t+1}. Καθώς το γ προσεγγίζει το 1, ο στόχος επιστροφής λαμβάνει περισσότερο υπόψη τις μελλοντικές ανταμοιβές και ο πράκτορας γίνεται πιο οξυδερκής.

Πολτικές και Συναρτήσεις Αξίας

Ένα από τα σημαντικότερα μεγέθη για την ενισχυτική μάθηση είναι η συνάρτηση τιμών ή αξίας (Value Function) η οποία συνήθως ορίζεται ως $V(t)$. Η συνάρτηση αυτή αντιπροσωπεύει την αξία μιας κατάστασης στην οποία μπορεί να βρεθεί ο πράκτορας. Η έννοια της αξίας ορίζεται από την άποψη των μελλοντικών ανταμοιβών ή, ακριβέστερα, από την άποψη της αναμενόμενης απόδοσης. Φυσικά οι ανταμοιβές που μπορεί να αναμένει ο πράκτορας στο μέλλον εξαρτώνται από τις ενέργειες που θα αναλάβει. Συνεπώς, οι συναρτήσεις αξίας ορίζονται σε σχέση με συγκεκριμένους τρόπους δράσης, που ονομάζονται *πολιτικές (policy)*.

Τυπικά, μια πολιτική είναι μια αντιστοίχιση από τις καταστάσεις σε πιθανότητες επιλογής κάθε πιθανής ενέργειας. Αν ο πράκτορας ακολουθεί την πολιτική π στη χρονική στιγμή t, τότε $\pi(a|s)$ είναι η πιθανότητα ότι $A_t = a$ αν $S_t = s$. Οι μέθοδοι ενισχυτικής μάθησης καθορίζουν τον τρόπο αλλαγής της πολιτικής του πράκτορα ως αποτέλεσμα της εμπειρίας του.

Η αξία μιας κατάστασης s, είναι η αναμενόμενη απόδοση όταν αρχίζουμε σε s και ακολουθούμε π από εκεί και πέρα. Για MDPs, μπορούμε να ορίσουμε υ_π επίσημα ως:

$$\upsilon_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] \;=\; \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]$$

όπου $\mathbb{E}_\pi[\,\cdot\,]$ δηλώνει την αναμενόμενη τιμή μιας τυχαίας μεταβλητής δεδομένου ότι ο πράκτορας ακολουθεί πολιτική π και t είναι οποιοδήποτε χρονικό βήμα. Σημειώστε ότι η τιμή της τερματικής κατάστασης, εάν υπάρχει, είναι πάντα μηδενική. Καλούμε τη συνάρτηση υ_π τη συνάρτηση κατάστασης-τιμής (state-value function) για την πολιτική π.

Ομοίως, ορίζουμε την αξία της ανάληψης δράσης a σε κατάσταση s σύμφωνα με μια πολιτική π, με συμβολισμό $q_\pi(s, a)$, ως την αναμενόμενη απόδοση που αρχίζει από s, λαμβάνοντας τη δράση a, και στη συνέχεια ακολουθώντας πολιτική π:

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \;=\; \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$$

Καλούμε τη συνάρτηση q_π τη συνάρτηση δράσης-τιμής (action-value function) για την πολιτική π. Οι συναρτήσεις τιμών υ_π και q_π μπορούν να εκτιμηθούν με τις εμπειρίες του πράκτορα. Για παράδειγμα, εάν ένας πράκτορας ακολουθεί την πολιτική π και διατηρεί έναν μέσο όρο, για κάθε κατάσταση που συναντάται, των

πραγματικών αποδόσεων που ακολούθησαν αυτήν την κατάσταση, τότε ο μέσος όρος θα συγκλίνει στην τιμή της κατάστασης, $\upsilon_\pi(s)$, καθώς ο αριθμός των φορών που συναντάται η κατάσταση προσεγγίζει το άπειρο. Ονομάζουμε μεθόδους εκτίμησης αυτού του είδους Monte Carlo, δηλαδή όταν περιλαμβάνουν τον υπολογισμό μέσης τιμής (και άλλων στατιστικών) πολλών τυχαίων δειγμάτων των πραγματικών αποδόσεων.

Μια θεμελιώδης ιδιότητα των συναρτήσεων αξίας που χρησιμοποιούνται στην ενισχυτική μάθηση είναι ότι ικανοποιούν αναδρομικές σχέσεις, βάσει των οποίων μπορούμε να χρησιμοποιήσουμε δυναμικό προγραμματισμό.

$$\begin{aligned} G_t \doteq & R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + ... \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + ...) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

$$\begin{aligned} \upsilon_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] \ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \sum_\alpha \pi(\alpha|s) \sum_{s',r} p(s', r|s, \alpha)[r + \gamma \upsilon_\pi(s')] \end{aligned}$$

Αυτή είναι η εξίσωση Bellman για υ_π. Εκφράζει την σχέση μεταξύ της αξίας μιας κατάστασης και των αξιών των διαδοχικών καταστάσεων της.

Βελτιστοποίηση πολιτικών και συναρτήσεων αξίας

Χρησιμοποιώντας τους παραπάνω ορισμούς μπορούμε να ορίσουμε , με ευθύ τρόπο, τη βέλτιστη πολιτική. Μια πολιτική π είναι καλύτερη από μια άλλη π' εάν το αναμενόμενο κέρδος είναι μεγαλύτερο για κάθε κατάσταση. Δηλαδή $\pi \geq \pi'$ αν $\upsilon_\pi(s) \geq \upsilon_{\pi'}(s),\ \forall s \in \mathcal{S}$. Υπάρχει πάντα μια βέλτιστη πολτική που είναι καλύτερη από τις άλλες. Μπορεί να είναι περισσότερες από μία και τις συμβολίζουμε π_* με την αντίστοιχη βέλτιστη συνάρτηση αξίας $\upsilon_*(s) \doteq \max_\pi \upsilon_\pi(s),\ \forall s \in \mathcal{S}$. Αντίστοιχα ορίζεται για τις συναρτήσεις δράσης-τιμής $q_*(s, a) \doteq \max_\pi {}_\pi(s, a),\ \forall s \in \mathcal{S}$ και $a \in \mathcal{A}$. Για το ζεύγος κατάστασης-δράσης (s, a), αυτή η συνάρτηση δίνει την αναμενόμενη απόδοση για ανάληψη δράσης α σε κατάσταση s και στη συνέχεια ακολουθώντας μια βέλτιστη πολιτική. Έτσι, μπορούμε να γράψουμε q_* σε συνάρτηση της υ_* ως εξής:

$$q_*(s, a) = \mathbb{E}[R_{t+1} + \gamma \upsilon_*(S_{t+1}) \mid S_t = s, A_t = a]$$

ησυνάρτηση τιμής βέλτιστης πολιτικής μπορεί επίσης να εκφραστεί συναρτήσει της δράσης-αξίας και εν συνέχεια να τις απομπλέξουμε ώστε να καταλήξουμε στην εύχρηστη μορφή Bellman.

$$\begin{aligned} \upsilon_*(s) = & \max_{\alpha \in \mathcal{A}(s)} q_*(s, \alpha) \\ = & \max_{\alpha \in \mathcal{A}(s)} \mathbb{E}[R_{t+1} + \gamma \upsilon_*(S_{t+1}) \mid S_t = s, A_t = a] \\ = & \max_{\alpha \in \mathcal{A}(s)} \sum_{s',r} p(s', r|s, a)[r + \gamma \upsilon_*(s')] \end{aligned} \tag{7}$$

$$
\begin{aligned}
q_*(s,a) &= \mathbb{E}[R_{t+1} + \gamma \max_{\alpha \in \mathcal{A}(s)} q_{\pi_*}(s,\alpha) |\ S_t = s, A_t = a] \\
&= \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{\alpha \in \mathcal{A}(s)} q_*(s',a)]
\end{aligned} \tag{8}
$$

2.3.3 Δυναμικός Προγραμματισμός

Ο όρος δυναμικός προγραμματισμός (DP) αναφέρεται σε μια συλλογή αλγορίθμων που μπορούν να χρησιμοποιηθούν για τον υπολογισμό των βέλτιστων πολιτικών που δίνουν ένα τέλειο μοντέλο περιβάλλοντος ως διαδικασία λήψης αποφάσεων Markov (MDP). Οι κλασσικοί αλγόριθμοι DP είναι περιορισμένης χρησιμότητας στη ενισχυτική μάθηση τόσο λόγω της υπόθεσής τους ενός τέλειου μοντέλου όσο και λόγω του μεγάλου αριθμού υπολογιστικών εξόδων τους, αλλά από θεωρητικής άποψης είναι ακόμη σημαντικές. Ο DP παρέχει μια ουσιαστική βάση για την κατανόηση των μεθόδων που παρουσιάζονται, οι οποίες μπορούν να θεωρηθούν ως προσπάθειες να επιτευχθεί το ίδιο αποτέλεσμα με τον DP, μόνο με λιγότερους υπολογισμούς και χωρίς να υποθέσουμε ένα τέλειο μοντέλο του περιβάλλοντος.

Οι αλγόριθμοι (DP) αποκτώνται μετατρέποντας τις εξισώσεις Bellman όπως τις 7 και 8 σε αναθέσεις, δηλαδή σε κανόνες ενημέρωσης για τη βελτίωση των προσεγγίσεων των επιθυμητών συναρτήσεων τιμής.

Αξιολόγηση και Βελτίωση Πολιτικής

Αρχικά εξετάζουμε πώς να υπολογίσουμε τη συνάρτηση αξίας κατάστασης υ_π για μια αυθαίρετη πολιτική π. Αυτό ονομάζεται αξιολόγηση πολιτικής (policy evaluation).

Algorithm 1: Iterative policy evaluation

Input π, the policy to be evaluated
repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $\upsilon \leftarrow V(s)$
    $V(s) \leftarrow \sum_\alpha \pi(\alpha|s) \sum_{s',r} p(s',r|s,a)\,[r + \gamma V(s')]$
    $\Delta \leftarrow max(\Delta, |\upsilon - V(s)|)$
until $\Delta < \theta$ *(a small positive number)*;
Output $V \approx \upsilon_\pi$

Ο λόγος για τον υπολογισμό της συνάρτησης αξίας μιας πολιτικής είναι να βοηθηθούμε στην εύρεση καλύτερων πολιτικών. Για να απαντήσουμε σε αυτή την ερώτηση ας εξετάσουμε το ενδεχόμενο επιλογής μιας δράσης α στην s και στην επόμενη συνεχίζουμε με την υπάρχουσα πολιτική π. Το βασικό κριτήριο είναι

εάν αυτό είναι μεγαλύτερο ή μικρότερο από $v_\pi(s)$. Αν όντως είναι μεγαλύτερη, προφανώς θα είναι καλύτερα να επιλέγουμε α κάθε φορά που βρισκόμαστε στην s και τότε η νέα πολιτική θα είναι συνολικά καλύτερη.

Το ότι αυτό είναι αλήθεια είναι μια ειδική περίπτωση ενός γενικού αποτελέσματος που ονομάζεται θεώρημα βελτίωσης πολιτικής.
Έστω π και π' ένα οποιοδήποτε ζεύγος ντετερμινιστικών πολιτικών έτσι ώστε:

$$q_\pi(s, \pi'(s)) \geq v_\pi(s), \quad \forall s \in \mathcal{S}$$

Τότε, η πολιτική π' πρέπει να είναι τόσο καλή ή και καλύτερη από την π. Δηλαδή, πρέπει να αποκτά μεγαλύτερη ή ίση αναμενόμενη επιστροφή από όλες τις καταστάσεις:

$$v_{\pi'}(s) \geq v_\pi(s), \quad \forall s \in \mathcal{S}$$

Μια φυσική επέκταση είναι να εξετάσουμε τις αλλαγές σε όλες τις καταστάσεις και ανάμεσα σε όλες τις πιθανές ενέργειες, επιλέγοντας σε κάθε κατάσταση τη δράση που δίνει καλύτερα αποτελέσματα σύμφωνα με το $q_\pi(s, \alpha)$.

$$\pi'(s) \doteq \arg\max_\alpha q_\pi(s, \alpha)$$

Η διαδικασία δημιουργίας μιας νέας πολιτικής που βελτιώνει την αρχική πολιτική, καθιστώντας την άπληστη ως προς την συνάρτηση αξίας-δράσης της αρχικής πολιτικής, καλείται βελτίωση της πολιτικής.

Επανάληψη Πολιτικής και συνάρτησης Αξίας

Μόλις μια πολιτική, π, βελτιωθεί χρησιμοποιώντας v_π για να δώσουμε μια καλύτερη πολιτική, π', τότε μπορούμε να υπολογίσουμε $v_{\pi'}$ και να την βελτιώσουμε ξανά για να έχουμε ακόμα καλύτερη π''. Μπορούμε έτσι να αποκτήσουμε μια ακολουθία μονοτονικών βελτιώσεων πολιτικών και συναρτήσεων αξίας.

Αυτός ο τρόπος εύρεσης μιας βέλτιστης πολιτικής ονομάζεται *επανάληψη πολιτικής*. Ένας πλήρης αλγόριθμος δίνεται στο επόμενο πλαίσιο. Σημειώστε ότι κάθε αξιολόγηση πολιτικής, η οποία είναι επαναληπτική διαδικασία, ξεκινά. με τη συνάρτηση αξίας για την προηγούμενη πολιτική. Αυτό συνήθως οδηγεί σε μεγάλη αύξηση της ταχύτητας σύγκλισης της αξιολόγησης πολιτικής (πιθανώς επειδή η συνάρτηση αξίας αλλάζει ελάχιστα από μία πολιτική στην επόμενη).

Algorithm 2: Policy iteration (using iterative policy evaluation)

Initialization $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$
- Policy Evaluation:
repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $\upsilon \leftarrow V(s)$
        $V(s) \leftarrow \sum_{s',r} p(s', r|s, a) [r + \gamma V(s')]$
        $\Delta \leftarrow max(\Delta, |\upsilon - V(s)|)$
until $\Delta < \theta$ *(a small positive number)*;
- Policy Improvement:
  policy-stable $\leftarrow$ *true*
  For each $s \in \mathcal{S}$:
      old-action $\leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a) [r + \gamma V(s')]$
      If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow false$
If *policy-stable* , stop and return $V \approx \upsilon_*$ and $\pi \approx \pi_*$; else go to 2

Ωστόσο η διαρκής εκτίμηση πολτικής εξακολουθεί να αποτελεί μειονέκτημα για την επανάληψη της πολιτικής καθώς μπορεί εν τέλει να είναι ένας παρατεταμένος επαναληπτικός υπολογισμός που απαιτεί πολλαπλές σαρώσεις του συνόλου των καταστάσεων. Αν η αξιολόγηση πολιτικής γίνει με επαναληπτικό τρόπο, τότε η σύγκλιση ακριβώς στο υ_π συμβαίνει μόνο στο όριο. Τέλεια ακρίβεια όσον αφορά την υ_π δεν είναι απαραίτητη, εάν η αναδυόμενη πολιτική είναι όντως σωστή. Ως εκ τούτου, το βήμα *αξιολόγησης πολιτικής* για την *επανάληψη πολιτικής* μπορεί να αποκοπεί με διάφορους τρόπους χωρίς να χάσει τις εγγυήσεις σύγκλισης της πολιτικής επανάληψης. Μια σημαντική ειδική περίπτωση είναι όταν η αξιολόγηση πολιτικής σταματά μετά από μία μόνο σάρωση (μία ενημέρωση για κάθε κατάσταση). Αυτός ο αλγόριθμος ονομάζεται *επανάληψη τιμής (Value iteration).*

Algorithm 3: Value iteration

Initialize array V arbitrarily
repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $\upsilon \leftarrow V(s)$
        $V(s) \leftarrow \max_\alpha \sum_{s',r} p(s', r|s, a) [r + \gamma V(s')]$
        $\Delta \leftarrow max(\Delta, |\upsilon - V(s)|)$
until $\Delta < \theta$ (a small positive number);
Output a deterministic policy, $\pi \approx \pi_*$, such that
    $\pi(s) = \arg\max_\alpha \sum_{s',r} p(s', r|s, a) [r + \gamma V(s')]$

2.3.4 Μέθοδοι Monte Carlo

Σε αυτό το κεφάλαιο εξετάζουμε τις πρώτες μεθόδους μάθησης για την εκτίμηση λειτουργιών αξίας και την ανακάλυψη βέλτιστων πολιτικών. Σε αντίθεση με το προηγούμενο κεφάλαιο, εδώ δεν έχουμε πλήρη γνώση του περιβάλλοντος. Οι μέθοδοι Monte Carlo απαιτούν μόνο ακολουθίες δειγμάτων εμπειρίας-γεγονότων, δράσεων και ανταμοιβών από πραγματική ή προσομοιωμένη αλληλεπίδραση με ένα περιβάλλον. Αν και απαιτείται ένα μοντέλο, το μοντέλο πρέπει να παράγει μόνο δείγματα μεταβάσεων, όχι τις πλήρεις κατανομές πιθανοτήτων όλων των δυνατών μεταβάσεων που απαιτούνται για τον δυναμικό προγραμματισμό (DP).

Οι μέθοδοι Monte Carlo δειγματοληπτούν και υπολογίζουν τον μέσο όρο των επιστροφών. Ενώ πριν υπολογίζαμε τις συναρτήσεις αξίας από τη γνώση του MDP, εδώ τις μαθαίνουμε από τις επιστροφές δειγμάτων με την MDP. Αρχικά θεωρούμε το πρόβλημα πρόβλεψης (τον υπολογισμό των v_π και q_π για μια σταθερή αυθαίρετη πολιτική π), στη συνέχεια την βελτίωση της πολιτικής και, τέλος, το πρόβλημα ελέγχου και τη λύση του.

Πρόβλεψη με Monte Carlo

Ξεκινάμε εξετάζοντας τις μεθόδους Monte Carlo για την εκμάθηση της συνάρτησης αξίας για μια συγκεκριμένη πολιτική. Υπενθυμίζουμε ότι η αξία μιας κατάστασης είναι η αναμενόμενη απόδοση - δηλαδή αναμενόμενη αθροιστική μελλοντική μειωμένη ανταμοιβή - ξεκινώντας από αυτή την κατάσταση. Ένας προφανής τρόπος για να εκτιμηθεί από την εμπειρία, λοιπόν, είναι απλώς ο μέσος όρος των αποδόσεων που παρατηρούνται μετά τις επισκέψεις σε αυτή τη κατάσταση. Καθώς παρατηρούνται περισσότερες αποδόσεις, ο μέσος όρος θα πρέπει να συγκλίνει στην αναμενόμενη τιμή. Σε αυτήν την ιδέα βασίζονται όλες οι μέθοδοι του Monte Carlo.

Algorithm 4: First-visit MC prediction, for estimating $V \approx v_\pi$

```
Initialize:
    π ← policy to be evaluated
    V ← an arbitrary state-value function
    Returns(s) ← an empty list, for all s ∈ S
repeat
    Generate an episode using π
    For each s in the episode:
        G ← the return that follows the first occurrence of s
        Append G to Return(s)
        V(s) ← average(Return(s))
until Forever;
```

Εάν το μοντέλο δεν είναι διαθέσιμο, τότε είναι ιδιαίτερα χρήσιμο να εκτιμηθούν οι τιμές δράσης (οι τιμές των ζευγών κατάστασης δράσης) και όχι οι τιμές κατάστασης. Με ένα μοντέλο, οι τιμές κατάστασης επαρκούν για τον προσδιορισμό μιας πολιτικής. Απλά κοιτάζουμε μπροστά ένα βήμα και επιλέγουμε όποια ενέργεια οδηγεί στον καλύτερο συνδυασμό ανταμοιβής και επόμενης κατάστασης. Χωρίς ένα μοντέλο, ωστόσο, οι αξίες κατάστασης και μόνο δεν επαρκούν. Πρέπει να εκτιμηθεί ρητά η αξία κάθε ενέργειας, προκειμένου οι τιμές να είναι χρήσιμες για την υποβολή μιας πολιτικής. Έτσι, ένας από τους πρωταρχικούς μας στόχους για τις μεθόδους Monte Carlo είναι να εκτιμήσουμε q_*. Επομένως, πρέπει πρώτα να εκτελέσουμε τον αλγόριθμο αξιολόγησης πολιτικής για τις τιμές δράσης.

Έλεγχος με Monte Carlo

Η βελτίωση της πολιτικής γίνεται κάνοντας την πολιτική άπληστη ως προς τη συνάρτηση της τρέχουσας αξίας. Σε αυτήν την περίπτωση έχουμε μια συνάρτηση δράσης-τιμής και συνεπώς δεν απαιτείται κανένα μοντέλο για την κατασκευή της άπληστης πολιτικής. Η βελτίωση της πολιτικής μπορεί στη συνέχεια να γίνει με την κατασκευή κάθε π_{k+1} ως την άπληστη πολιτική σε σχέση με το q_{π_k}. Όπως συζητήσαμε στο προηγούμενο κεφάλαιο, το θεώρημα μας διαβεβαιώνει ότι κάθε π_{k+1} είναι καλύτερο ή εξίσου καλό με π_k. Αυτό μας διαβεβαιώνει ότι η συνολική διαδικασία συγκλίνει στη βέλτιστη συνάρτηση πολιτικής και βέλτιστης τιμής. Με αυτόν τον τρόπο οι μέθοδοι του Monte Carlo μπορούν να χρησιμοποιηθούν για να βρεθούν οι βέλτιστες πολιτικές που δίδονται μόνο σε δείγματα επεισοδίων και με καμία άλλη γνώση της δυναμικής του περιβάλλοντος.

Algorithm 5: Monte Carlo ES(Exploring Starts),for estimating $\pi \approx \pi_*$

Initialize, for all $s \in \mathcal{S}, \alpha \in \mathcal{A}(s)$:
    $Q(s, \alpha) \leftarrow$ arbitrary
    $Returns(s, \alpha) \leftarrow$ an empty list
    $\pi(s) \leftarrow$ arbitrary
repeat
    Choose $S_0 \in S$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability > 0
    Generate an episode starting from S_0, A_0, and then using π
    For each s, α in the episode:
        $G \leftarrow$ the return that follows the first occurrence of s, α
        Append G to $Return(s, \alpha)$
        $Q(s, \alpha) \leftarrow$ average$(Return(s, \alpha))$
    For each s in the episode:
        $\pi(s) \leftarrow \arg\max_\alpha Q(s, \alpha)$
until *Forever*;

Υπάρχει ένα λεπτό σημείο που πρέπει να αναφερθεί σχετικά με αυτόν τον αλγόριθμο. Στην αρχή κάθε επεισοδίου επιλέγεται τυχαία ένα ζευγάρι δράσης και κατάστασης, πράγμα που σημαίνει ότι υποθέτουμε ότι ολόκληρος ο χώρος των καταστάσεων-δράσεων είναι διαθέσιμος από την αρχή - εξού και η ονομασία

exploring starts. Δεδομένης της δεύτερης παραδοχής απεριόριστων επεισοδίων, εγγυόμαστε, τελικά, την επίσκεψη σε ολόκληρο τον χώρο καταστάσεων. Αυτό δεν είναι πρακτικά εφικτό.

Ο μόνος γενικός τρόπος για να διασφαλιστεί ότι όλες οι ενέργειες επιλέγονται απεριόριστα συχνά είναι για τον πράκτορα να συνεχίσει να τις επιλέγει. Υπάρχουν δύο προσεγγίσεις για να διασφαλιστεί αυτό, οι μέθοδοι on-policy και off-policy. Οι on-policy μέθοδοι προσπαθούν να αξιολογήσουν ή να βελτιώσουν την πολιτική που χρησιμοποιείται για τη λήψη αποφάσεων, ενώ οι μέθοδοι off-policy αξιολογούν ή βελτιώνουν μια πολιτική διαφορετική από αυτήν που χρησιμοποιείται για τη δημιουργία των δεδομένων. Η μέθοδος Monte Carlo ES που αναπτύχθηκε πιο πάνω είναι ένα παράδειγμα μιας μεθόδου on-policy.

Τώρα παρουσιάζουμε πώς μπορεί να σχεδιαστεί μια μέθοδος ελέγχου Monte Carlo για την πολιτική που δεν χρησιμοποιεί την μη ρεαλιστική υπόθεση των exploring starts.

Στις πολιτικές ελέγχου on-policy η πολιτική είναι γενικά 'soft', που σημαίνει ότι $\pi(\alpha|s) > 0$ για όλα τα $s \in S$ και όλα τα $\alpha \in \mathcal{A}(s)$, αλλά σταδιακά μετατοπίζεται όλο και πιο κοντά σε μια αιτιοκρατική βέλτιστη πολιτική. Η on-policy μέθοδος που παρουσιάζουμε εδώ χρησιμοποιεί ε-άπληστες πολιτικές, που σημαίνει ότι τις περισσότερες φορές επιλέγουν μια ενέργεια που έχει μέγιστη εκτιμώμενη τιμή δράσης, αλλά με πιθανότητα ε επιλέγουν τυχαία.

Δηλαδή, όλες οι μη άπληστες ενέργειες έχουν την ελάχιστη πιθανότητα επιλογής, $\frac{\epsilon}{|\mathcal{A}(s)|}$, και το υπόλοιπο μέρος της πιθανότητας, $1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|}$, δίνεται στην άπληστη δράση.

Algorithm 6: On-policy first-visit MC control (for ϵ-soft policies), estimates $\pi \approx \pi_*$

Initialize, for all $s \in \mathcal{S}, \alpha \in \mathcal{A}(s)$:
    $Q(s, \alpha) \leftarrow$ arbitrary
    $Returns(s, \alpha) \leftarrow$ an empty list
    $\pi(\alpha|s) \leftarrow$ an arbitrary ϵ-soft policy
repeat
    Generate an episode using π
    For each s, α in the episode:
        $G \leftarrow$ the return that follows the first occurrence of s, α
        Append G to $Return(s, \alpha)$
        $Q(s, \alpha) \leftarrow$ average($Return(s, \alpha)$)
    For each s in the episode:
        $A^* \leftarrow \arg\max_\alpha Q(s, \alpha)$
        For all $\alpha \in \mathcal{A}(s)$:
        $$\pi(\alpha|s) \leftarrow \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|}, & \text{if } \alpha = A^* \\ \frac{\epsilon}{\mathcal{A}(s)}, & \text{if } \alpha \neq A^* \end{cases}$$
until *Forever*;

Όλες οι μέθοδοι ελέγχου αντιμετωπίζουν ένα δίλημμα: Επιδιώκουν να μάθουν

τις αξίες δράσης δεδομένης βέλτιστης συμπεριφοράς, αλλά πρέπει να συμπεριφέρονται μη βέλτιστα για να εξερευνήσουν όλες τις δράσεις (ώστε να βρουν τις βέλτιστες ενέργειες).

Η on-policy προσέγγιση που ακολουθήσαμε προηγουμένως είναι ένας συμβιβασμός - μαθαίνει τιμές δράσης όχι για τη βέλτιστη πολιτική αλλά για μια σχεδόν βέλτιστη πολιτική που εξακολουθεί να εξερευνά. Μια πιο απλή προσέγγιση είναι να χρησιμοποιήσουμε δύο πολιτικές, μία για την οποία μαθαίνουμε και η οποία γίνεται η βέλτιστη πολιτική και μια που εξερευνά και χρησιμοποιείται για τη δημιουργία της βέλτιστης. Η πολιτική που μαθαίνουμε ονομάζεται πολιτική στόχου και η πολιτική που χρησιμοποιείται για τη δημιουργία συμπεριφοράς ονομάζεται πολιτική συμπεριφοράς. Αυτή είναι η διαδικασία που ονομάσαμε off-policy. Σχεδόν όλες οι μέθοδοι off-policy χρησιμοποιούν τη δειγματοληψία σημασίας (importance sampling), μια γενική τεχνική για την εκτίμηση των αναμενόμενων τιμών ύπο μία κατανομή που δίνεται σε δείγματα από μια άλλη.

Algorithm 7: Off-policy MC control, for estimating $\pi \approx \pi_*$

Initialize, for all $s \in \mathcal{S}, \alpha \in \mathcal{A}(s)$:
    $Q(s, \alpha) \leftarrow$ arbitrary
    $C(s, \alpha) \leftarrow 0$
    $\pi(s) \leftarrow \arg\max_\alpha Q(S_t, \alpha)$
repeat
    $b \leftarrow$ any soft policy
    Generate an episode using b:
        $S_0, A_0, R_1, ..., S_{T-1}, A_{T-1}, R_T, S_T$
    $G \leftarrow 0$
    $W \leftarrow 1$
    For $t = T-1, T-2, ..., 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)}[G - Q(S_t, A_t)$
        $\pi(s) \leftarrow \arg\max_\alpha Q(S_t, \alpha)$
        If $A_t \neq \pi(s)$ then exit For loop
        $W \leftarrow W \frac{1}{b(A_t|S_t)}$
until *Forever*;

2.3.5 Μέθοδοι Χρονικών διαφορών

Η μάθηση με TD είναι ένας συνδυασμός ιδεών του Monte Carlo και ιδεών δυναμικού προγραμματισμού DP. Όπως οι μέθοδοι Monte Carlo, οι μέθοδοι TD μπορούν να μάθουν απευθείας από την ακατέργαστη εμπειρία χωρίς ένα μοντέλο της δυναμικής του περιβάλλοντος. Όπως και το DP, οι μέθοδοι TD επικαιροποιούν εκτιμήσεις που βασίζονται εν μέρει σε άλλες, χωρίς να περιμένουν ένα τελικό αποτέλεσμα (bootstrap).

Πρόβλεψη με Χρονικές Διαφορές (Temporal Difference)

Η πιο απλή μέθοδος της οικογένειας αυτής, TD(0) αξιολογεί την πολιτική ως εξής:

Algorithm 8: Tabular TD(0) for estimating υ_π

```
Input: the policy π to be evaluated
Initialize V(s) arbitrarily
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        A ← action given by π for S
        Take action A, observe R, S'
        V(S) ← V(S) + α[R + γV(S') − V(S)]
        S ← S'
    until S is terminal
```

Επειδή η TD(0) βασίζει την ενημέρωσή της εν μέρει σε μια υπάρχουσα εκτίμηση, λέμε ότι πρόκειται για μια μέθοδο bootstrapping, όπως η DP. Οι μέθοδοι TD χρησιμοποιούν ουσιαστικά την προαναφερθείσα bootstrapping τεχνική του DP αλλά αντί των πιθανών μεταβάσεων που παρέχονται από τη μέθοδο που βασίζεται σε μοντέλο, χρησιμοποιεί την ιδέα δειγματοληψίας παρόμοια με την MC.

Να σημειωθεί ότι η ποσότητα στην παρένθεση, στην ενημέρωση TD(0) είναι ένα είδος σφάλματος, που μετράει τη διαφορά μεταξύ της εκτιμώμενης τιμής $V(S_t)$ και της καλύτερης εκτίμησης $R_{t+1} + \gamma V(S_{t+1})$. Αυτή η ποσότητα, που ονομάζεται TD-error (σφάλμα), προκύπτει σε διάφορες μορφές καθ'όλη την ενισχυτική μάθηση:

$$\delta_t \doteq R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \tag{9}$$

Προφανώς, οι μέθοδοι TD έχουν ένα πλεονέκτημα έναντι των μεθόδων DP επειδή δεν απαιτούν ένα μοντέλο περιβάλλοντος, των κατανομών πιθανότητας και της επόμενης κατάστασης. Το επόμενο πιο προφανές πλεονέκτημα των μεθόδων TD μέσω των μεθόδων Monte Carlo είναι ότι αυτές υλοποιούνται με φυσικό τρόπο σε μια πλήρως εντατική μορφή. Με τις μεθόδους Monte Carlo κάποιος πρέπει να περιμένει μέχρι το τέλος ενός επεισοδίου, επειδή μόνο τότε είναι γνωστή η επιστροφή, ενώ με τις μεθόδους TD πρέπει να περιμένει μόνο ένα χρονικό βήμα.

Εκπληκτικά συχνά αυτό αποδεικνύεται κρίσιμο. Ορισμένες εφαρμογές έχουν πολύ μεγάλα επεισόδια, έτσι ώστε η καθυστέρηση όλων των μαθημάτων μέχρι το τέλος του επεισοδίου είναι πολύ αργή. Άλλες εφαρμογές είναι συνεχείς εργασίες και δεν έχουν καθόλου επεισόδια. Τέλος, όπως σημειώσαμε στο προηγούμενο κεφάλαιο, μερικές μέθοδοι του Monte Carlo πρέπει να αγνοούν ή να κάνουν έκπτωση σε επεισόδια για τα οποία γίνονται πειραματικές ενέργειες, πράγμα που μπορεί να επιβραδύνει σημαντικά την εκμάθηση. Οι μέθοδοι TD είναι πολύ λιγότερο ευαίσθητες σε αυτά τα προβλήματα επειδή μαθαίνουν από κάθε μετάβαση ανεξάρτητα από τις ενέργειες που ακολουθούν.

Έλεγχος με Χρονικές Διαφορές (Temporal Difference)

Γυρναμε τώρα στη χρήση μεθόδων πρόβλεψης TD για το πρόβλημα ελέγχου. Ως συνήθως, ακολουθούμε το πρότυπο της γενικευμένης επανάληψης πολιτικής (GPI), μόνο αυτή τη φορά χρησιμοποιώντας μεθόδους TD για το κομμάτι αξιολόγησης ή πρόβλεψης. Όπως συμβαίνει με τις μεθόδους του Monte Carlo, αντιμετωπίζουμε την ανάγκη να ανταλλάξουμε την εξερεύνηση και την εκμετάλλευση (exploration-exploitation) και οι προσεγγίσεις πέφτουν ξανά σε δύο κύριες κατηγορίες: στην on-policy και off-policy. Ξεκινάμε με την on-policy.

Algorithm 9: Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Initialize $Q(s,a)$ for all $s \in S, a \in A(s)$, arbitrarily, and $Q(terminal - state, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q
    $Q(S,A) \leftarrow Q(S,A) + a[R + \gamma Q(S',A') - Q(S,A)$
    $S \leftarrow S', A \leftarrow A'$
  until S is terminal

Προχωρούμε τώρα σε μια πολύ σημαντική ανακάλυψη στο πλαίσιο της RL και έναν ακρογωνιαίο λίθο του σύγχρονου RL και αυτής τη εργασίας, του αλγορίθμου ελέγχου TD off policy, γνωστού ως Q-Learning, ο οποίος ορίζεται ως:

Algorithm 10: Q-learning (off-policy TD control)
for estimating $\pi \approx \pi_*$

Initialize $Q(s, a)$ for all $s \in S, a \in A(s)$, arbitrarily, and
$Q(terminal - state, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat for each step of episode:
    Choose A from S using policy derived from Q
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + a[R + \gamma \max_\alpha Q(S', \alpha) - Q(S, A)$
    $S \leftarrow S$
  until S is terminal

Σε αυτή την περίπτωση, η συνάρτηση δράσης-τιμής που μαθαίνουμε, Q, προσεγγίζει άμεσα q_*, τη βέλτιστη συνάρτηση δράσης-τιμής, ανεξάρτητα από την ακολουθούμενη πολιτική. Αυτό απλουστεύει δραματικά την ανάλυση του αλγορίθμου και επέτρεψε από νωρίς την ανακάλυψη αποδείξεων περί σύγκλισης. Η πολιτική εξακολουθεί να επιδρά στο αποτέλεσμα, διότι καθορίζει ποια ζεύγη καταστάσεων-δράσεων επισκέπτονται και ενημερώνονται.

Εκκίνιση ν-βημάτων(n-step bootstrapping)

Σε αυτό το κεφάλαιο ενοποιούμε τις μεθόδους Monte Carlo (MC) και τις μεθόδους χρονικής διαφοράς (TD) ενός σταδίου που παρουσιάζονται στα προηγούμενα δύο κεφάλαια. Ούτε οι μέθοδοι MC ούτε οι μέθοδοι TD ενός βήματος είναι πάντα οι καλύτερες. Σε αυτό το κεφάλαιο παρουσιάζουμε n-step TD μεθόδους που γενικεύουν και τις δύο μεθόδους έτσι ώστε να μπορεί κάποιος να μετατοπίζεται ομαλά από την μία στην άλλη όπως απαιτείται για να ικανοποιήσει τις απαιτήσεις μιας συγκεκριμένης εργασίας. Οι μέθοδοι n-βημάτων καλύπτουν ένα φάσμα με τις μεθόδους MC στη μία άκρη και με τη μέθοδο TD ενός βήματος στο άλλο. Οι καλύτερες μέθοδοι είναι συχνά ενδιάμεσες μεταξύ των δύο άκρων.

Υποθέτουμε πως θέλουμε να εκτιμήσουμε την v_π από δείγματα επεισοδίων που δημιουργούνται χρησιμοποιώντας π. Οι μέθοδοι του Monte Carlo πραγματοποιούν μια ενημέρωση για κάθε κατάσταση με βάση ολόκληρη την ακολουθία παρατηρημένων ανταμοιβών από αυτήν την κατάσταση μέχρι το τέλος του επεισοδίου. Η ενημέρωση των μεθόδων TD ενός βήματος, από την άλλη πλευρά, βασίζεται μόνο στην επόμενη ανταμοιβή, ξεκινώντας από την αξία της κατάστασης ένα βήμα αργότερα ως υποκατάστατο για τα υπόλοιπα οφέλη. Ένα είδος ενδιάμεσης μεθόδου θα πραγματοποιούσε μια ενημέρωση βασισμένη σε έναν ενδιάμεσο αριθμό ανταμοιβών: περισσότερες από μία, αλλά λιγότερες από όλες μέχρι την λήξη.

Οι μέθοδοι που χρησιμοποιούν ενημερώσεις n-βημάτων εξακολουθούν να είναι μέθοδοι TD, επειδή εξακολουθούν να αλλάζουν μια προηγούμενη εκτίμηση που βασίζεται στο πώς διαφέρει από μια μεταγενέστερη εκτίμηση. Τώρα, η μεταγενέστερη εκτίμηση δεν είναι ένα βήμα αργότερα, αλλά n βήματα αργότερα. Οι μέθοδοι στις οποίες η χρονική διαφορά επεκτείνεται σε n βήματα ονομάζονται n-step TD μέθοδοι. Οι μέθοδοι TD που συζητήθηκαν στη προηγούμενη παράγραφο χρησιμοποίησαν όλες τις ενημερώσεις ενός βήματος, γι΄ αυτό τις ονομάζουμε μεθόδους TD ενός βήματος.

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}) \quad (10)$$

Ονομάζουμε αυτήν την ποσότητα ως στόχο της ενημέρωσης. Οι δείκτες στο $G_{t:t+n}$ δείχνουν ότι είναι μια αποκομμένη επιστροφή για το χρόνο t χρησιμοποιώντας ανταμοιβές μέχρι τη χρονική στιγμή $t+n$. Θα το χρησιμοποιήσουμε ως μέρος ενός από τους αλγορίθμους μας αργότερα στη πορεία.

3 Βαθιά Ενισχυτική Μάθηση

3.1 Εισαγωγή

Η μάθηση για τον έλεγχο των πρακτόρων άμεσα από αισθητήρια δεδομένα υψηλών διαστάσεων, όπως όραση και λόγος, είναι μια από τις μακρόχρονες προκλήσεις της ενίσχυτικής μάθησης (RL). Οι πιο επιτυχημένες εφαρμογές RL που λειτουργούν σε αυτούς τους τομείς έχουν βασιστεί σε χειροποίητα χαρακτηριστικά που συνδυάζονται με γραμμικές συναρτήσεις αξίας ή αναπαραστάσεις πολιτικής. Είναι σαφές ότι η απόδοση τέτοιων συστημάτων βασίζεται σε μεγάλο βαθμό στην ποιότητα της παράστασης χαρακτηριστικών. Οι πρόσφατες εξελίξεις στη βαθιά μάθηση έχουν καταστήσει δυνατή την εξαγωγή χαρακτηριστικών υψηλού επιπέδου από ακατέργαστα αισθητήρια δεδομένα, οδηγώντας σε ανακαλύψεις στην όραση υπολογιστή και αναγνώριση ομιλίας. Αυτές οι μέθοδοι χρησιμοποιούν μια σειρά αρχιτεκτονικών νευρωνικών δικτύων, συμπεριλαμβανομένων των συνελικτικών δικτύων, πολλαπλών στρώσεων perceptrons, περιορισμένων μηχανών Boltzmann και επαναλαμβανόμενα νευρωνικά δίκτυα και έχουν εκμεταλλευτεί τόσο την επίβλεπόμενη όσο και την μη επιβλεπόμενη μάθηση. Φαίνεται φυσικό να αναρωτηθούμε ,πρώτον , αν παρόμοιες τεχνικές θα μπορούσαν επίσης να είναι επωφελείς για την RL με αισθητήρια δεδομένα και να εφαρμόσουν αυτούς τους αλγορίθμους σε σημεία αναφοράς (benchmarks) όπως στην περίπτωση μας τα βιντεοπαιχνίδια και σε δεύτερο λόγο για το αν αυτά μπορούν να χρησιμοποιηθούν για τη μεταφορά της μάθησης (transfer learning). Παρόλο που η απάντηση είναι θετική, δεν είναι τόσο απλή όσο απαιτεί προσεκτική εξέταση του τρόπου συνδυασμού των νευρωνικών δικτύων και των αλγορίθμων RL.

Σε αυτό το κεφάλαιο παρουσιάζουμε τη θεωρία πίσω από δύο οικογένειες σύγχρονων RL αλγορίθμων που ξεκίνησαν από το ζευγάρωμα της βαθιάς μάθησης και RL. Η πρώτη οικογένεια αλγορίθμων είναι η συνέχεια της Q Learning, την οποία έχουμε ήδη συζητήσει, και η δεύτερη είναι αυτή των policy gradients (μτφ. κλίσεις πολιτκών).

3.2 Deep Q-Networks

3.2.1 DQN αλγόριθμος

Η ενισχυτική μάθηση παρουσιάζει πολλές προκλήσεις από την προοπτική βαθιάς μάθησης. Πρώτον, οι πιο επιτυχημένες εφαρμογές βαθιάς μάθησης απαιτούν μεγάλα ποσά δεδομένων εκπαίδευσης, σημαδεμένα με ταμπέλες με το χέρι. Οι αλγόριθμοι RL, από την άλλη πλευρά, πρέπει να είναι σε θέση να μάθουν από ένα βαθμωτό σήμα ανταμοιβής που είναι συχνά αραιό, θορυβώδες και καθυστερημένο. Η καθυστέρηση μεταξύ των ενεργειών και των ανταμοιβών που προκύπτουν, που μπορεί να είναι χιλιάδες χρονικά διαστήματα, φαίνεται ιδιαίτερα αποθαρρυντική σε σύγκριση με την άμεση συσχέτιση μεταξύ των δεδομένων και των στόχων που εντοπίζονται στην εποπτευόμενη μάθηση. Ένα άλλο ζήτημα είναι ότι οι περισσότεροι αλγόριθμοι deep-learning υποθέτουν ότι τα δείγματα δεδομένων είναι

ανεξάρτητα, ενώ στην ενισχυτική μάθηση συναντά κανείς συνήθως ακολουθίες υψηλά συσχετισμένων καταστάσεων. Επιπλέον, στην RL η κατανομή δεδομένων αλλάζει καθώς ο αλγόριθμος μαθαίνει νέες συμπεριφορές, το οποίο μπορεί να είναι προβληματικό για μεθόδους βαθιάς μάθησης που υποθέτουν μια σταθερή υποκείμενη κατανομή.

Αυτός ο αλγόριθμος καταδεικνύει ότι ένα συνελικτικό νευρωνικό δίκτυο μπορεί να ξεπεράσει αυτές τις προκλήσεις για να μάθει επιτυχημένες πολιτικές ελέγχου από ακατέργαστα δεδομένα βίντεο σε σύνθετα περιβάλλοντα RL. Το δίκτυο είναι εκπαιδευμένο με μια παραλλαγή του αλγορίθμου Q-learning. Η αρχική εργασία το εκπαιδεύει με stochastic gradient descent (SGD) για να ενημερώσει τα βάρη, αν και πιο σύγχρονες παραλλαγές χρησιμοποιούν τον Adam.

Αυτό είναι το σημείο όπου τα επόμενα θέματα που συζητούνται στην ενότητα 2 συνενώνονται σε ένα ενοποιημένο πλαίσιο. Το πρώτο πράγμα που πρέπει να θυμόμαστε είναι πώς η Q Learning προσεγγίζει την βέλτιστη λύση $Q^*(s,a)$ και πως δημιουργούμε έναν στόχο για τις τρέχουσες εκτιμήσεις μας με την εξίσωση Bellman: $Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}}[r + \gamma \max_{\alpha'} Q^*(s', \alpha') | s, \alpha)]$,

Αναφέραμε μόνο την πινακοειδή (tabular) μέθοδο μέχρι στιγμής όπου για κάθε κατάσταση αποθηκεύουμε ξεχωριστή τιμή δράσης χωρίς γενίκευση. Στην πράξη, αυτή η βασική προσέγγιση είναι εντελώς ανέφικτη. Αντ' αυτού, είναι συνηθισμένο να χρησιμοποιούμε μια συνάρτηση - προσέγγιση για να υπολογίσουμε τη συνάρτηση τιμής-δράσης, στην περίπτωση μας ένα νευρωνικό δίκτυο, παραμετροποιημένο ως $Q(s, \alpha; \boldsymbol{\theta}) \approx Q^*(s, \alpha)$,

Έχουμε ήδη συζητήσει πώς εκπαιδεύεται ένα νευρωνικό δίκτυο, ελαχιστοποιώντας μια συνάρτηση απώλειας μέσω της gradient descend και back-propagation. Στο πλαίσιο των δικτύων Q, η εν λόγω συνάρτηση απώλειας είναι το μέσο τετραγωνικό σφάλμα όπου ο στόχος δίνεται από την εξίσωση Bellman για την συνάρτηση δράσης-αξίας Q:

$$L_i(\boldsymbol{\theta}_i) = \mathop{\mathbb{E}}_{s, \alpha \sim \rho(\cdot)} [(y_i - Q(s, \alpha; \boldsymbol{\theta}_i))^2] \tag{11}$$

όπου $y_i = \mathbb{E}_{s' \sim \mathcal{E}}[r + \gamma \max_{\alpha'} Q(s', \alpha; \boldsymbol{\theta}_{i-1}) | s, \alpha]$ είναι ο στόχος στην επανάληψη i και $\rho(s, \alpha)$ είναι μια κατανομή πιθανότητας σε σχέση με τις ακολουθίες s, α και την αναφέρουμε ως κατανομή συμπεριφοράς. Οι παράμετροι από την προηγούμενη επανάληψη $\boldsymbol{\theta}_{i-1}$ διατηρούνται σταθερές κατά τη βελτιστοποίηση της συνάρτησης απώλειας $L_i(\boldsymbol{\theta}_i)$. Σημειώστε ότι οι στόχοι εξαρτώνται από τα βάρη του δικτύου. Αυτό έρχεται σε αντίθεση με τους στόχους που χρησιμοποιούνται για την επιβλεπόμενη μάθηση, οι οποίοι καθορίζονται πριν αρχίσει η μάθηση. Παραγωγίζοντας τη συνάρτηση απώλειας σε σχέση με τα βάρη φθάνουμε στην ακόλουθη έκφραση για την κλίση της:

$$\nabla L_i(\boldsymbol{\theta}_i) = \mathop{\mathbb{E}}_{s, \alpha \sim \rho(\cdot); s \sim \mathcal{E}} [(y_i - Q(s, \alpha; \boldsymbol{\theta}_i)) \nabla_{\boldsymbol{\theta}_i} Q(s, \alpha; \boldsymbol{\theta}_i)] \tag{12}$$

Το γεγονός ότι ο στόχος βασίζεται στο ίδιο δίκτυο που εκπαιδεύεται έρχεται με τις δικές του παγίδες. Ένα σημαντικό συστατικό του αλγορίθμου DQN είναι η χρήση ενός δεύτερου δικτύου στόχου, πανομοιότυπο με το εκπαιδευόμενο. Η ενσωμάτωση αυτού του τεχνάσματος μπορεί να προκαλέσει σύγχυση στην αρχή,

ωστόσο χωρίς αυτό ο στόχος μεταβάλλεται διαρκώς, λόγω του ότι βασίζεται εν μέρει από το ίδιο Q Network που εκπαιδεύεται. Το δίκτυο στόχος, με τις παραμέτρους $\boldsymbol{\theta}^-$, είναι το ίδιο με το online δίκτυο, με την εξαίρεση ότι οι παράμετροί του αντιγράφονται σε κάθε τ βήμα από το online δίκτυο, έτσι ώστε $\boldsymbol{\theta}^- = \boldsymbol{\theta}_t$ και διατηρούνται σταθερά σε όλα τα άλλα βήματα. Ο στόχος που χρησιμοποιείται από το DQN είναι τότε $y_t = R_{t+1} + \gamma \max Q(S_{t+1}, \alpha; \boldsymbol{\theta}^-)$ (13). Έτσι για αυτά τα βήματα η συμπεριφορά του στόχου, στην εξίσωση απώλειας, παραμένει σταθερή.

Τέλος ο αλγόριθμος χρησιμοποιεί μια τεχνική γνωστή ως επανάληψη εμπειρίας (experience replay) όπου αποθηκεύουμε τις εμπειρίες του πράκτορα σε κάθε χρονικό βήμα, $e_t = (s_t, a_t, r_t, s_{t+1})$ σε ένα σύνολο δεδομένων $\mathcal{D} = e_1, ..., e_N$, συγκεντρωμένα έπειτα από πολλά επεισόδια σε μια μνήμη επανάληψης. Κατά τη διάρκεια του εσωτερικού βρόχου του αλγορίθμου, εφαρμόζουμε ενημερώσεις $Q-Learning$ ή ενημερώσεις $mini-batch$ σε δείγματα εμπειρίας, $e \sim \mathcal{D}$, που επιλέγεται τυχαία από τη μνήμη. Αφού πραγματοποιηθεί η επανάληψη της εμπειρίας, ο πράκτορας επιλέγει και εκτελεί μια ενέργεια σύμφωνα με μια πολιτική $\epsilon-greedy$. Ο πλήρης αλγόριθμος, τον οποίο καλούμε $Deep\ Q-learning$, παρουσιάζεται εδώ:

Algorithm 11: Deep Q-learning with Experience Replay

Initialize replay memory D to capacity N.
Initialize action-value function Q with random weights
for episode = 1, M **do**
  Initialize sequence $s_1 = x_1$ and preprocess sequence $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action α_t
    otherwise select $\alpha_t = \max_\alpha Q^*(\phi(s_t), \alpha; \boldsymbol{\theta})$
    Execute α_t in emulator (environment) and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, \alpha_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, \alpha_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{\alpha'} Q(\phi_{j+1}, \alpha'; \boldsymbol{\theta}^-) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \boldsymbol{\theta}))^2$ according to equation 12
    If (episode-1) $\cdot\, T + t \equiv 0 \mod \tau$ then $\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$
  end for
end for

Αυτή η προσέγγιση έχει αρκετά πλεονεκτήματα σε σχέση με την τυπική online Q-learning. Πρώτον, κάθε βήμα της εμπειρίας μπορεί να χρησιμοποιηθεί σε πολλές ενημερώσεις βάρους, γεγονός που επιτρέπει μεγαλύτερη αποτελεσματικότητα των δεδομένων. Δεύτερον, η απευθείας εκμάθηση από διαδοχικά δείγματα είναι αναποτελεσματική, λόγω των ισχυρών συσχετισμών μεταξύ των δειγμάτων. Η τυχαιοποίηση των δειγμάτων σπάει αυτές τις συσχετίσεις και επομένως μειώνει

τη διακύμανση των ενημερώσεων. Τρίτον, κατά την εκμάθηση στην πολιτική, οι τρέχουσες παράμετροι καθορίζουν το επόμενο δείγμα δεδομένων στο οποίο εκπαιδεύονται οι παράμετροι. Για παράδειγμα, εάν η ενέργεια μεγιστοποίησης είναι να μετακινηθείτε αριστερά τότε τα δείγματα εκπαίδευσης θα κυριαρχούνται από δείγματα από την αριστερή πλευρά. αν η ενέργεια μεγιστοποίησης μεταβιβάζεται στη συνέχεια δεξιά, τότε η κατανομή εκπαίδευσης θα αλλάξει επίσης. Είναι εύκολο να δούμε πώς μπορεί να προκύψουν ανεπιθύμητοι βρόχοι ανάδρασης και ότι οι παράμετροι θα μπορούσαν να κολλήσουν σε κακή τοπική ελάχιστη τιμή ή ακόμα και να αποκλίνουν καταστροφικά. Με τη χρήση επανάληψης εμπειρίας (experience replay), η κατανομή συμπεριφοράς υπολογίζεται κατά μέσον όρο σε πολλές από τις προηγούμενες καταστάσεις, εξομαλύνοντας τη μάθηση και αποφεύγοντας ταλαντώσεις ή αποκλίσεις στις παραμέτρους. Σημειώστε ότι όταν μαθαίνουμε από την επανάληψη της εμπειρίας, είναι απαραίτητο να μάθουμε off-policy (επειδή οι τρέχουσες παράμετροί μας είναι διαφορετικές από εκείνες που χρησιμοποιούνται για τη δημιουργία του δείγματος), γεγονός που υποκινεί την επιλογή της Q-learning.

Ο παραπάνω αλγόριθμος χρησιμεύει ως θεμέλιο της οικογένειας αλγορίθμων DQN που συζητάμε και κορυφώνεται με τον Rainbow τον οποίο συζητούμε στο τέλος της ενότητας.

3.2.2 Διπλό DQN (Double DQN)

Η Q-learning είναι γνωστό ότι μερικές φορές μαθαίνει μη ρεαλιστικά υψηλές τιμές δράσης επειδή περιλαμβάνει ένα βήμα μεγιστοποίησης επιλέγοντας ανάμεσα από τις εκτιμώμενες τιμές δράσης, το οποίο τείνει να προτιμά υπερεκτιμημένες από υποτιμημένες τιμές. Οι υπερεκτιμήσεις οφείλονται σε ανεπαρκή ευελιξία στη λειτουργία και θόρυβο. Ωστόσο, αργότερα παρουσιάστηκε [7], ότι υπερεκτίμηση μπορεί να συμβεί όταν οι τιμές δράσης είναι ανακριβείς, ανεξάρτητα από την πηγή του σφάλματος προσέγγισης. Φυσικά, οι ανακριβείς εκτιμήσεις αξίας είναι ο κανόνας κατά τη διάρκεια της μάθησης, γεγονός που δείχνει ότι οι υπερτιμήσεις μπορεί να είναι πολύ πιο συχνές από ό,τι προηγουμένως εκτιμήθηκε.

Είναι ανοιχτό το ερώτημα , εάν υπάρξουν υπερεκτιμήσεις,εάν αυτό επηρεάζει αρνητικά τις επιδόσεις στην πράξη. Οι υπεραισιόδοξες εκτιμήσεις αξίας δεν είναι απαραίτητα ένα πρόβλημα από μόνο του. Εάν όλες οι τιμές είναι ομοιόμορφα υψηλότερες, τότε διατηρούνται οι προτιμήσεις σχετικών ενεργειών και δεν θα περίμενε κανείς ότι η προκύπτουσα πολιτική να ήταν χειρότερη. Επιπλέον, είναι γνωστό ότι μερικές φορές είναι καλό να είμαστε αισιόδοξοι: η αισιοδοξία ενόψει της αβεβαιότητας είναι μια πολύ γνωστή τεχνική εξερεύνησης. Εάν, ωστόσο, οι υπερεκτιμήσεις δεν είναι ομοιόμορφες και δεν συγκεντρωθούν σε καταστάσεις για τις οποίες επιθυμούμε να μάθουμε περισσότερα, τότε ενδέχεται να επηρεάσουν αρνητικά την ποιότητα της πολιτικής που προκύπτει.

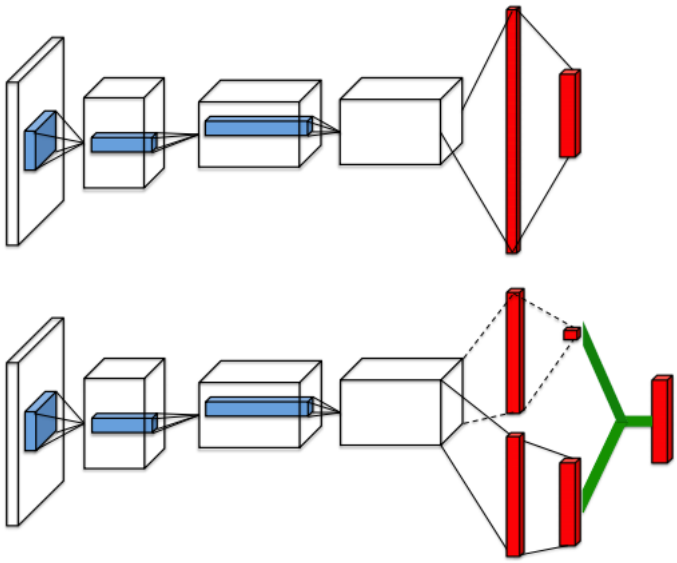
Για να αποφευχθεί αυτό, μπορούμε να αποσυνδέσουμε την επιλογή δράσης από την αξιολόγησή της. Αν και δεν είναι πλήρως αποσυνδεδεμένο, το δίκτυοστόχος της αρχιτεκτονικής DQN παρέχει ένα φυσικό υποψήφιο για τη δεύτερη συνάρτηση τιμής, χωρίς να χρειάζεται να εισάγει πρόσθετα δίκτυα. Προτείνουμε

λοιπόν να αξιολογήσουμε την άπληστη πολιτική σύμφωνα με το online δίκτυο, αλλά χρησιμοποιώντας το δίκτυο στόχο για να εκτιμήσουμε την αξία του. Η ενημέρωσή του είναι ίδια με αυτή για το DQN, αλλά αντικαθιστά το στόχο (13) με:

$$y_t = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_{\alpha} Q(S_{t+1}, \alpha; \boldsymbol{\theta}); \boldsymbol{\theta}^-) \tag{14}$$

Αυτή η έκδοση του Double DQN είναι ίσως η ελάχιστη πιθανή αλλαγή στο DQN προς τον Double Q-learning. Ο στόχος είναι να αποκομιστεί το μεγαλύτερο μέρος του Double-learning, διατηρώντας παράλληλα τον υπόλοιπο αλγόριθμο DQN άθικτο με ελάχιστη υπολογιστική επιβάρυνση.

3.2.3 Αρχιτεκτονική Μονομαχίας (Dueling DQN)



Η έρευνα εστιάζεται κυρίως στο σχεδιασμό βελτιωμένων αλγορίθμων ελέγχου και RL ή απλώς στην ενσωμάτωση των υφιστάμενων αρχιτεκτονικών νευρωνικών δικτύων σε μεθόδους RL. Εδώ, υιοθετούμε μια εναλλακτική αλλά συμπληρωματική προσέγγιση που επικεντρώνεται πρωτίστως στην καινοτομία μιας αρχιτεκτονικής νευρωνικών δικτύων που είναι πιο κατάλληλη για RL χωρίς μοντέλα.

Η προτεινόμενη αρχιτεκτονική δικτύου, η οποία ονομάζουμε αρχιτεκτονική μονομαχίας dueling architecture, διαχωρίζει ρητά την αναπαράσταση των τιμών καταστάσεων και των (εξαρτώμενων από την κατάσταση) πλεονεκτημάτων δράσης. Η αρχιτεκτονική dueling αποτελείται από δύο ροές που αντιπροσωπεύουν τις συναρτήσεις αξίας και πλεονεκτήματος, ενώ μοιράζονται μια κοινή μονάδα βασισμένη σε συνελικτικά δίκτυα για την εξαγωγή χαρακτηριστικών. Τα δύο ρεύματα συν-

δυάζονται μέσω ενός ειδικού στρώματος συσσωμάτωσης για την παραγωγή μιας εκτίμησης της συνάρτησης Q της τιμής κατάστασης δράσης.

Όπως θα δούμε ξανά με τη δεύτερη οικογένεια αλγορίθμων RL (δηλαδή policy gradients), υπάρχει μια μακρά ιστορία συναρτήσεων πλεονεκτημάτων.

Διαισθητικά, η αρχιτεκτονική μονομαχίας μπορεί να μάθει ποιες καταστάσεις είναι (ή όχι) πολύτιμες, χωρίς να χρειάζεται να μάθουμε την επίδραση κάθε δράσης για κάθε κατάσταση. Αυτό είναι ιδιαίτερα χρήσιμο σε καταστάσεις όπου οι ενέργειές του δεν επηρεάζουν το περιβάλλον με κανένα σχετικό τρόπο.

Για να γίνει αυτό, πρέπει πρώτα να ορίσουμε αυτή τη νέα σημαντική ποσότητα, τη λειτουργία πλεονεκτήματος, που σχετίζεται με τις αξίες και τις συναρτήσεις Q μιας δεδομένης πολιτικής π:

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

Διαισθητικά, η συνάρτηση τιμών V μετράει πόσο καλό είναι να είναι σε μια συγκεκριμένη κατάσταση. Η συνάρτηση Q, ωστόσο, μετρά την τιμή της επιλογής μιας συγκεκριμένης ενέργειας όταν βρισκόμαστε σε αυτή την κατάσταση. Η συνάρτηση πλεονεκτήματος αφαιρεί την τιμή της κατάστασης από τη συνάρτηση Q για να αποκτήσει ένα σχετικό μέτρο της σημασίας κάθε δράσης.

Χρησιμοποιώντας τον ορισμό του πλεονεκτήματος, θα μπορούσαμε να μπούμε στον πειρασμό να κατασκευάσουμε τη συνιστώσα συγκέντρωσης ως εξής:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + A(s,a;\theta,\alpha), \tag{15}$$

Ωστόσο, πρέπει να έχουμε κατά νου ότι η $Q(s,a;\theta,\alpha,\beta)$ είναι μόνο μια παραμετροποιημένη εκτίμηση της πραγματικής συνάρτησης Q. Επιπλέον, θα ήταν λάθος να καταλήξουμε στο συμπέρασμα ότι το $V(s;\theta,\beta)$ είναι ένας καλός εκτιμητής της συνάρτησης κατάστασης-τιμής ή ομοίως ότι το $A(s,a;\theta,\alpha)$ παρέχει εύλογη εκτίμηση της συνάρτησης πλεονεκτήματος. Η εξίσωση 15 δεν μπορεί να προσδιοριστεί με την έννοια ότι με το Q δεν μπορούμε να ανακτήσουμε μοναδικά το V και το A. Για να αντιμετωπιστεί αυτό το ζήτημα της αναγνωρισιμότητας (identifiability), μπορούμε να αναγκάσουμε τον εκτιμητή συνάρτησης πλεονεκτήματος να έχει μηδενικό πλεονέκτημα στην επιλεγμένη ενέργεια. Δηλαδή αφήνουμε την τελευταία μονάδα του δικτύου να εφαρμόσει την προς τα εμπρός αντιστοίχιση:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + A(s,a;\theta,\alpha) - \max_{a' \in |\mathcal{A}|} A(s,a';\theta,\alpha) \tag{16}$$

Τώρα, για $a^* = \arg\max_{a' \in \mathcal{A}} Q(s,a';\theta,\alpha,\beta) = \arg\max_{a' \in \mathcal{A}} A(s,a';\theta,\alpha)$ παίρνουμε $Q(s,a^*;\theta,\alpha,\beta) = V(s;\theta,\beta)$ Επομένως, το ρεύμα $V(s;\theta,\beta)$ παρέχει μια εκτίμηση της συνάρτησης τιμών, ενώ το άλλο ρεύμα παράγει μια εκτίμηση της συνάρτησης πλεονεκτήματος.

Μια εναλλακτική μονάδα συσσώρευσης αντικαθιστά τον max τελεστή με έναν μέσο όρο:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + A(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|} \sum_{a' \in |\mathcal{A}|} A(s,a';\theta,\alpha) \tag{17}$$

Είναι σημαντικό να σημειωθεί ότι η εξίσωση 17 αντιμετωπίζεται και εφαρμόζεται ως μέρος του δικτύου και όχι ως ξεχωριστό αλγοριθμικό βήμα. Η εκπαίδευση

των αρχιτεκτονικών μονομαχιών, όπως συμβαίνει με το τυποποιημένο δίκτυο Q, απαιτεί μόνο οπίσθια διάδοση. Οι εκτιμήσεις $V(s;\theta,\beta)$ και $A(s,a;\theta,\alpha)$ υπολογίζονται αυτόματα χωρίς καμία επιπλέον επίβλεψη ή αλγοριθμικές τροποποιήσεις.

3.2.4 Επαναφορά εμπειρίας με προτεραιότητα (Prioritized Experience Replay)

Η επανάληψη της εμπειρίας επιτρέπει στους εκπαιδευτικούς πράκτορες online ενίσχυσης να θυμούνται και να επαναχρησιμοποιούν εμπειρίες από το παρελθόν. Μέχρι στιγμής, οι μεταβάσεις εμπειρίας δειγματοληπτούνται ομοιόμορφα από τη μνήμη επανάληψης. Ωστόσο, αυτή η προσέγγιση απλά αναπαράγει μεταβάσεις με την ίδια συχνότητα που είχαν αρχικά βιώσει, ανεξάρτητα από τη σημασία τους. Τώρα προχωρούμε στη χρήση επαναλαμβανόμενης εμπειρίας με προτεραιότητα.

Έχουμε ήδη συζητήσει πώς ένας online RL πράκτορας που παρατηρεί μια ροή εμπειριών και μαθαίνει από αυτή καθώς εμφανίζεται, αντιμετωπίζει δύο σημαντικά προβλήματα: (α) ισχυρά συσχετισμένες ενημερώσεις στα βάρρη που διασπούν την i.i.d παραδοχή πολλών δημοφιλών στοχαστικών αλγορίθμων που βασίζονται σε κλίση και (β) την ταχεία λησμόνηση πιθανών σπάνιων εμπειριών που θα ήταν χρήσιμες αργότερα.

(Σημ. :Στη θεωρία των πιθανοτήτων και στις στατιστικές, μια συλλογή τυχαίων μεταβλητών είναι ανεξάρτητη και ταυτόσημα κατανεμημένη εάν κάθε τυχαία μεταβλητή έχει την ίδια κατανομή πιθανότητας με τις άλλες και όλες είναι ανεξάρτητες. Αυτή η ιδιότητα συνήθως συντομεύεται ως i.i.d. Στη θεωρία της μηχανικής μάθησης, η υπόθεση i.i.d. συχνά γίνεται για σύνολα δεδομένων εκπαίδευσης που υποδηλώνουν ότι όλα τα δείγματα παράγονται από την ίδια διαδικασία και πως αυτή υποτίθεται ότι δεν έχει μνήμη προηγούμενων παραγόμενων δειγμάτων.)

Η επανάληψη της εμπειρίας αντιμετωπίζει και τα δύο αυτά ζητήματα: με την εμπειρία που είναι αποθηκευμένη σε μια μνήμη επανάληψης, γίνεται δυνατό να σπάσουμε τις χρονικές συσχετίσεις με την ανάμιξη όλο και λιγότερο πρόσφατων εμπειριών για τις ενημερώσεις και η σπάνια εμπειρία θα χρησιμοποιηθεί για περισσότερες από μία ενημερώσεις.

Δεδομένου ότι δειγματοληπτούμε ομοιόμορφα από την μνήμη, εμπειρίες που εμφανίζονται συχνά έχουν μεγαλύτερη πιθανότητα επανάληψης. Η επανάληψη προτεραιότητας επιτρέπει τους πράκτορες να εξετάσουν μεταβάσεις ανεξάρτητα της συχνότητας που τις βιώνουν. Η βασική ιδέα είναι ότι ένας πράκτορας RL μπορεί να μάθει πιο αποτελεσματικά από μερικές μεταβάσεις παρά από άλλες. Οι μεταβάσεις μπορεί να είναι περισσότερο ή λιγότερο περίεργες, περιττές ή συναφείς με τις εργασίες. Ορισμένες μεταβάσεις μπορεί να μην είναι άμεσα χρήσιμες στον πράκτορα, αλλά μπορεί να γίνουν όταν η ικανότητα του πράκτορα αυξηθεί.

Το κεντρικό στοιχείο της επανάληψης προτεραιότητας είναι το κριτήριο με το οποίο μετράται η σημασία κάθε μετάβασης. Ένα εξιδανικευμένο κριτήριο θα είναι το ποσό που ο πράκτορας RL μπορεί να μάθει από μια μετάβαση στην τρέχουσα κατάσταση του (αναμενόμενη πρόοδος της μάθησης). Ενώ αυτό το μέτρο δεν είναι άμεσα προσπελάσιμο, ένα εύλογο αντικατάστατο είναι το μέγεθος του σφάλματος TD της μετάβασης, το οποίο υποδεικνύει πόσο απροσδόκητη είναι η μετάβαση:

συγκεκριμένα, πόσο μακριά είναι η αξία από την εκτίμησή του επόμενου βήματος. Αυτό είναι ιδιαίτερα κατάλληλο για αυξητικούς, online RL αλγόριθμους , όπως SARSA ή Q-learning, που ήδη υπολογίζουν το σφάλμα TD και ενημερώνουν τις παραμέτρους ανάλογα με το δ. Σε ορισμένες περιπτώσεις το σφάλμα TD μπορεί να είναι μια κακή εκτίμηση, π.χ. όταν οι ανταμοιβές είναι θορυβώδεις.

Μια προφανής προσέγγιση με βάση το εν λόγω σφάλμα είναι να υποθέσουμε έναν αλγόριθμο «άπληστου TD-σφάλματος προτεραιότητας». Αυτός ο αλγόριθμος αποθηκεύει το τελευταίο σφάλμα TD που συναντάμε μαζί με κάθε μετάβαση στη μνήμη επανάληψης. Η μετάβαση με το μεγαλύτερο απόλυτο σφάλμα TD επαναλαμβάνεται από τη μνήμη. Μια ενημερωμένη έκδοση Q-learning εφαρμόζεται σε αυτήν τη μετάβαση, η οποία ενημερώνει τα βάρη ανάλογα με το σφάλμα TD. Νέες μεταβάσεις φτάνουν χωρίς ένα γνωστό TD-σφάλμα, γι αύτό το θέτουμε με τη μέγιστη προτεραιότητα, προκειμένου να διασφαλίσουμε ότι όλες οι εμπειρίες θα εξεταστούν τουλάχιστον μία φορά.

Ωστόσο, η άπληστη προτεραιότητα με βάση το TD-σφάλμα έχει αρκετά προβλήματα. Πρώτον, για να αποφευχθούν -χρονικά ακριβές- σαρώσεις σε ολόκληρη τη μνήμη επαναλήψεων, τα σφάλματα TD ενημερώνονται μόνο για τις μεταβάσεις που επαναλαμβάνονται. Μια συνέπεια είναι ότι οι μεταβάσεις που έχουν χαμηλό TD σφάλμα κατά την πρώτη επίσκεψη μπορεί να μην επαναληφθούν για μεγάλο χρονικό διάστημα (πράγμα που σημαίνει ότι δεν είναι ποτέ αποτελεσματικό με μια επαναλαμβανόμενη μνήμη συρόμενου παραθύρου). Περαιτέρω, είναι ευαίσθητο σε αιχμές θορύβου (π.χ. όταν οι ανταμοιβές είναι στοχαστικές), οι οποίες μπορούν να επιδεινωθούν με εκτόξευση, όπου τα σφάλματα προσέγγισης εμφανίζονται ως άλλη πηγή θορύβου. Τέλος, η άπληστη προτεραιότητα επικεντρώνεται σε ένα μικρό υποσύνολο της εμπειρίας: τα σφάλματα συρρικνώνονται αργά, ειδικά όταν χρησιμοποιούνται προσεγγιστικές συναρτήσεις, πράγμα που σημαίνει ότι οι αρχικά υψηλές μεταβάσεις σφαλμάτων παίρνονται συχνά για επανάληψη. Αυτή η έλλειψη ποικιλομορφίας καθιστά το σύστημα επιρρεπές σε over-fitting.

Για να ξεπεραστούν αυτά τα ζητήματα, εισάγουμε μια στοχαστική μέθοδο δειγματοληψίας που παρεμβάλλεται μεταξύ καθαρής προτεραιότητας άπληστων και ομοιόμορφης δειγματοληψίας. Διασφαλίζουμε η πιθανότητα να είναι μονοτονική σε προτεραιότητα μετάβασης, ενώ εγγυάται μια μη μηδενική πιθανότητα ακόμη και για τη μετάβαση με τη χαμηλότερη προτεραιότητα. Συγκεκριμένα, ορίζουμε την πιθανότητα μετάβασης δειγματοληψίας i ως:

$$P(i) = \frac{p_i^a}{\sum_k p_k^a} \tag{18}$$

Ο εκθέτης α καθορίζει πόση προτεραιότητα χρησιμοποιείται, με $a = 0$ που αντιστοιχεί στην ομοιόμορφη περίπτωση. Η πρώτη παραλλαγή που θεωρούμε είναι η άμεση, αναλογική προτεραιότητα όπου $p_i = |\delta_i| + \epsilon$. Η δεύτερη παραλλαγή είναι μια έμμεση ιεράρχηση προτεραιότητας όπου $p_i = \frac{1}{rank(i)}$ όπου $rank(i)$ είναι η τάξη της μετάβασης i όταν η μνήμη επαναλήψεως ταξινομείται σύμφωνα με το δ_i. Και οι δύο κατανομές είναι μονοτονικές στο $|\delta|$, αλλά το τελευταίο είναι πιθανότερο να είναι πιο ανθεκτικό, καθώς δεν είναι ευαίσθητο στις υπερβολικές τιμές.

Υλοποίηση. Η αναλογική παραλλαγή δέχεται μια αποτελεσματική υλοπο-

ίηση που βασίζεται σε μια δομή δεδομένων (sum-tree) (όπου κάθε κόμβος είναι το άθροισμα των παιδιών του, με τις προτεραιότητες ως κόμβοι-φύλλα), τα οποία μπορούν να ενημερωθούν και να δειγματοληφθούν αποτελεσματικά.

Η εκτίμηση της αναμενόμενης τιμής με στοχαστικές ενημερώσεις βασίζεται σε αυτές τις ενημερώσεις που αντιστοιχούν στην ίδια κατανομή με την προσδοκία τους. Η αναπαραγωγή με προτεραιότητα εισάγει προκατάληψη επειδή αλλάζει αυτή την κατανομή με ανεξέλεγκτο τρόπο και ως εκ τούτου αλλάζει τη λύση με την οποία οι εκτιμήσεις θα συγκλίνουν (ακόμα και αν η πολιτική και η κατανομή των καταστάσεων είναι σταθερές). Μπορούμε να διορθώσουμε αυτή την προκατάληψη χρησιμοποιώντας τα βάρη της δειγματοληψίας (IS).

$$w_i = \left(\frac{1}{N \cdot P(i)}\right)^{\beta}$$

Αυτά τα βάρη μπορούν να αναδιπλωθούν στην ενημερωμένη έκδοση Q-learning χρησιμοποιώντας $w_i\delta_i$ αντί για δ_i. Στα τυπικά σενάρια ενισχυτικής μάθησης, η αμερόληπτη φύση των ενημερώσεων είναι πιο σημαντική κοντά στην σύγκλιση στο τέλος της εκπαίδευσης, καθώς η διαδικασία είναι κατά κύριο λόγο μη στάσιμη ούτως ή άλλως, λόγω των μεταβαλλόμενων πολιτικών, των κατανομών των καταστάσεων και των στόχων bootstrap. Υποθέτουμε ότι μια μικρή προκατάληψη μπορεί να αγνοηθεί σε αυτό το πλαίσιο. Επομένως εκμεταλλευόμαστε την ευελιξία της ανόπτησης του ποσού της διόρθωσης importance-sampling με την πάροδο του χρόνου, καθορίζοντας ένα χρονοδιάγραμμα στον εκθέτη β που φθάνει στο 1 μόνο στο τέλος της μάθησης. Στην πράξη, ανασυνδυάζουμε γραμμικά β από την αρχική τιμή β_0 στο 1. Σημειώστε ότι η επιλογή αυτού του υπερπαραμετρικού στοιχείου αλληλεπιδρά με την επιλογή εκθέτη προτεραιότητας α· αυξάνοντας ταυτόχρονα την προτεραιότητα της δειγματοληψίας πιο επιθετικά ταυτόχρονα με τη διόρθωσή της πιο έντονα.

Algorithm 12: Double DQN with proportional prioritization

Input: minibatch k, step-size η, replay period K and size N , exponents α and β, budget T.
Initialize replay memory $\mathcal{H} = \emptyset.\Delta = 0, p_1 = 1$
Observe S_0 and choose $A_0 \sim \pi_\theta(S_0)$
for $t = 1$ **to** T **do**
  Observe $\mathrm{S}_t, R_t, \gamma_t$
  Store transition $(S_{t-1}, A_{t-1}, R_t, \gamma, S_t)$ in $\mathcal{H}$ with maximal priority
  $p_t = \max_{i<t} p_i$
  if $t \equiv 0 \mod K$ **then**
    for $j = 1$ **to** k **do**
      Sample transition $j \sim P(j) = p_j^a / \sum_k p_k^a$
      Compute importance-sampling weight
      $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$
      Compute TD-error:
      $\delta_j = R_j + \gamma Q_{target}(S_j, \arg\max_a Q(S_j, a)) - Q(S_{j-1}, A_{j-1})$
      Update transition priority: $p_j \leftarrow |\delta_j|$
      Accumulate weight-change $\Delta \leftarrow \Delta + w_j \dot{\delta}_j \cdot \nabla_\theta Q(S_{j-1}, A_{j-1})$
    end for
    Update weights $\theta \leftarrow \theta + \eta \cdot \Delta$, reset $\Delta = 0$
    From time to time copy weights into target network $\theta_{target} \leftarrow \theta$
  end if
  Choose action $A_t \sim \pi_\theta(S_t)$
end for

3.2.5 Θορυβώδη δίκτυα για εξερεύνηση (Noisy Networks for exploration)

Ένα άλλο σημαντικό θέμα που δεν έχει εξεταστεί είναι το θέμα της εξερεύνησης. Οι περισσότερες στρατηγικές εξερεύνισης βασίζονται σε τυχαίες διαταραχές της πολιτικής του πράκτορα, όπως η ϵ-άπληστη μέθοδος, την οποία έχουμε ήδη αναφέρει και χρησιμοποιήσει, ή την κανονικοποίηση της εντροπίας, για να προκαλέσουν νέες συμπεριφορές. Εντούτοις, τέτοιες τοπικές διαταραχές που προκαλούν δυσχέρειες είναι απίθανο να οδηγήσουν σε πρότυπα συμπεριφοράς μεγάλης κλίμακας που απαιτούνται για αποτελεσματική εξερεύνηση σε πολλά περιβάλλοντα.

Εδώ προτείνεται μια απλή εναλλακτική προσέγγιση, που ονομάζεται NoisyNet [;], όπου μαθημένες διαταραχές των βαρών του δικτύου χρησιμοποιούνται για να οδηγήσουν την εξερεύνηση.

Η βασική ιδέα είναι ότι μία και μόνο αλλαγή στο διάνυσμα βάρους μπορεί να προκαλέσει μια συνεπή και δυνητικά πολύ περίπλοκη, εξαρτώμενη από την κατάσταση αλλαγή της πολιτικής σε πολλαπλά χρονικά βήματα - σε αντίθεση με τις προηγούμενες προσεγγίσεις, όπου σε κάθε βήμα προστίθεται στην πολιτική ασυσχέτιστος θόρυβος (και στην περίπτωση του ϵ-άπληστου, ανεξάρτητος των καταστάσεων). Οι διαταραχές λαμβάνουν δείγματα από τη διανομή θορύβου. Η

διακύμανση της διαταραχής είναι μια παράμετρος που μπορεί να θεωρηθεί ως η ενέργεια του εγχυμένου θορύβου. Αυτές οι παράμετροι διακύμανσης μαθαίνονται χρησιμοποιώντας gradients από τη συνάρτηση απώλειας, παράλληλα με τις άλλες παραμέτρους του πράκτορα.

Πιο συγκεκριμένα, έστω $y = f_\theta(x)$ ένα νευρωνικό δίκτυο παραμετροποιημένο με ένα διάνυσμα θορυβώδων παραμέτρων θ, που παίρνει είσοδο x και δίνει έξοδο y. Εκφράζουμε τις θορυβώδεις παραμέτρους θ ως $\theta = \mu + \Sigma \bigodot \epsilon$, όπου $\zeta = (\mu, \Sigma)$ είναι ένα σύνολο διανυσμάτων των μαθητών παραμέτρων, ϵ είναι ένα διάνυσμα μηδενικού μέσου θορύβου με σταθερές στατιστικές, και $\bigodot$ αντιπροσωπεύει πολλαπλασιασμό ανά στοιχείο.

Η συνήθης απώλεια του νευρωνικού δικτύου τυλίγεται από τον τελεστή προσδοκώμενης τιμής ως προς τον θόρυβο $\epsilon : \bar{L}(\zeta) = \mathbb{E}[L(\theta)]$. Η βελτιστοποίηση τώρα συμβαίνει ως προς το σύνολο των παραμέτρων ζ. Για παράδειγμα ας θεωρήσουμε ένα απλό γραμμικό στρώμα ενός νευρωνικού δικτύου με εισόδους p και εξόδους q , που αντιπροσωπεύονται από την εξίσωση : $y = wx + b$, όπου $x \in \mathbb{R}^p$, $w \in \mathbb{R}^{q \times p}$. Τότε , το αντίστοιχο θορυβώδες γραμμικό στρώμα ορίζεται ως:

$$y = (\mu^w + \sigma^w \bigodot \epsilon^w)x + \mu^b + \sigma^b \bigodot \epsilon^b, \tag{19}$$

όπου $\mu^w + \sigma^w \bigodot \epsilon^w$ και $\mu^b + \sigma^b \bigodot \epsilon^b$ αντικαθιστούν τα w και b αντίστοιχα. Οι παράμετροι $\mu^w, \sigma^w \in \mathbb{R}^{q \times p}$ και $\mu^b, \sigma^b \in \mathbb{R}^q$ μαθαίνονται ενώ οι ϵ^w, ϵ^b είναι τυχαίες μεταβλητές θορύβου.

Στρέφουμε την προσοχή μας τώρα σε συγκεκριμένες περιπτώσεις κατανομής θορύβου για γραμμικά στρώματα σε θορυβώδες δίκτυο. Εξετάζουμε δύο επιλογές: Ανεξάρτητος θόρυβος Gauss, ο οποίος χρησιμοποιεί ανεξάρτητο Gaussian θόρυβο εισόδου ανά βάρος, και παραγοντοποιημένο Gaussian θόρυβο, ο οποίος χρησιμοποιεί ανεξάρτητο θόρυβο μόνο ανά έξοδο και ανά είσοδο. Ο κύριος λόγος για τη χρήση παραγοντοποιημένου θορύβου Gaussian είναι για να μειώσουμε τον χρόνο υπολογισμού της δημιουργίας τυχαίων αριθμών στους αλγορίθμους μας. Αυτή η υπολογιστική επιβάρυνση είναι ιδιαίτερα απαγορευτική στην περίπτωση των ενός-thread πρακτόρων όπως το DQN και το Dueling DQN.

Οι δύο επιλογές αναλυτικά:

(a) Ανεξάρτητος θόρυβος Gauss: ο θόρυβος που εφαρμόζεται σε κάθε βάρος και μεροληψία είναι ανεξάρτητος, όπου κάθε καταχώρηση $\epsilon^w_{i,j}$ (αντίστοιχα κάθε ϵ^b_j) της τυχαίας μήτρας ϵ^w (αντίστοιχα του τυχαίου ϵ^b) προέρχεται από μια μοναδιαία κατανομή Gauss. Αυτό σημαίνει ότι για κάθε θορυβώδες γραμμικό στρώμα, υπάρχουν $\pi\chi + \chi$μεταβλητές θορύβου (για εισόδους π στο στρώμα και χ εξόδους).

(b) Παραγοντοποιημένος θόρυβος Gauss: παραγοντοποιόντας τα $\epsilon^w_{i,j}$, μπορούμε να χρησιμοποιήσουμε p μοναδιαίες *Gaussian* μεταβλητές ϵ_i για θόρυβο των εισόδων και και q μοναδιαίες *Gaussian* μεταβλητές ϵ_j για θόρυβο των εξόδων (έτσι $p + q$ μοναδιαίες *Gaussian* μεταβλητές συνολικά). Κάθε $\epsilon^w_{i,j}, \epsilon_j$ μπορεί τότε να γραφτεί ως:

$$\epsilon^w_{i,j} = f(\epsilon_i) f(\epsilon_j), \tag{20}$$

$$\epsilon^b = f(\epsilon_j) \tag{21}$$

όπου φ είναι μια πραγματική συνάρτηση. Ένα παράδειγμα που χρησιμοποιήθηκε είναι $f(x) = sgn(x)|x|$.

3.2.6 Ενισχυτική Μάθηση με κατανομή (Distributional Reinforcement Learning)

Αυτή τη φορά εξετάζουμε τη θεμελιώδη σημασία της κατανομής της αξίας: η κατανομή της τυχαίας απόδοσης που λαμβάνεται από έναν πράκτορα ενισχυτικής μάθησης. Αυτό έρχεται σε αντίθεση με τις προαναφερθείσες προσεγγίσεις οι οποίες υποδηλώνουν την προσδοκία αυτής της απόδοσης ή της αξίας.

Συγκεκριμένα χρησιμοποιήσαμε την εξίσωση Bellman για την αναμενόμενη χρησιμότητα ενός ζεύγους κατάστασης δράσης Q για την καθοδήγηση του πράκτορα:

$$Q(s,a) = R + \gamma\, \mathbb{E}[Q(s,a)] \tag{22}$$

Αυτή τη φορά προτείνεται μια προοπτική που αφορά την ίδια την κατανομή της χρησιμότητας. Συγκεκριμένα, ο κύριος στόχος της μελέτης μας είναι η ίδια η τυχαία επιστροφή, Z, της οποίας η αναμενόμενη τιμή είναι η τιμή Q. Αυτή η τυχαία απόδοση περιγράφεται επίσης από μια αναδρομική εξίσωση, αλλά αυτή αφορά κατανομές:

$$Z(s,a) = R(s,a) + \gamma Z(S', A') \tag{23}$$

Πιο συγκεκριμένα, η επιστροφή Z^π είναι το άθροισμα των εκπτωτικών ανταμοιβών κατά μήκος της τροχιάς των αλληλεπιδράσεων του πράκτορα με το περιβάλλον, με το Q^π να είναι η αναμενόμενη τιμή. Θα μοντελοποιήσουμε την κατανομή της αξίας χρησιμοποιώντας μια διακριτή κατανομή που παραμετροποιείται από $N \in \mathbb{N}$, και $V_{min}, V_{max} \in \mathbb{R}$ που είναι η ελάχιστη και μέγιστη τιμή και το ενδιάμεσο τμήμα κατανεμημένο εξίσου στα άτομα N -το σύνολο των οποίων που ονομάζουμε στήριξη της κατανομής- και ορίζεται ως:
$\{z_i = V_{min} + i\Delta z : 0 \leq i < N\}, \Delta z := \frac{V_{max} - V_{min}}{N-1}$. Κατά μία έννοια, αυτά τα άτομα είναι οι κανονικές επιστροφές της διανομής μας. Οι πιθανότητες ατόμων δίδονται από ένα παραμετρικό μοντέλο $\theta : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}^N$

$$Z_\theta(s,a) = z_i \quad \text{w.p} \quad p_i(s,a) := \frac{e^{\theta_i(s,a)}}{\sum_j e^{\theta_j(s,a)}} \tag{24}$$

Η διακριτή κατανομή έχει τα πλεονεκτήματα ότι είναι εξαιρετικά εκφραστική και φιλική προς τον υπολογισμό. Δειγματολειπτούμε και προβάλλουμε την ενημέρωση Bellman $\hat{\mathcal{T}} Z_\theta$ στην στήριξη του Z_θ, ανάγωντας την ενημέρωση του Bellman σε ταξινόμηση πολλών κλάσεων (multi class classification). Έστω π η άπληστη πολιτική ως προς $\mathbb{E}[Z_\theta]$. Με δεδομένη μια μετάβαση δείγματος (s, a, r, s'), υπολογίζουμε την ενημέρωση του Bellman $\hat{\mathcal{T}} z_j := r + \gamma z_j$ για κάθε άτομο z_j, στη συνέχεια κατανέμουμε την πιθανότητα $p_j(s', \pi(s'))$ στους άμεσους γείτονες της $\hat{\mathcal{T}} z_j$. Το i-οστό συστατικό της προβαλλόμενης ενημέρωσης $\Phi \hat{\mathcal{T}} Z_\theta(s,a)$ είναι:

$$\Phi\hat{\mathcal{T}}Z_\theta(s,a)_i = \sum_{j=0}^{N-1}\left[1 - \frac{|[\hat{\mathcal{T}}z_j]_{V_{min}}^{V_{max}} - z_i|}{\Delta z}\right]_0^1 p_j(s', \pi(s')) \tag{25}$$

όπου $[\cdot]_a^b$ περιορίζει το όρισμα ανάμεσα στο $[a, b]$. Ως συνήθως αναπαριστούμε την κατανομή της επόμενης κατάστασης μέσω μιας διαφορετικής προσεγγιστικής συνάρτησης με σταθερές παραμέτρους θ' που ενημερώνεται περιοδικά ($\theta' \leftarrow \theta$ όπως πριν). Αυτή τη φορά η απώλεια $L_{s,a}(j)$ είναι ο όρος ¨διασταυρούμενης εντροπίας της απόκλισης KL' (cross-entropy term of the KL divergence):

$$D_{KL}(\Phi\hat{\mathcal{T}}Z_{\bar{\theta}}(s,a)||Z_\theta(s,a)) \tag{26}$$

Ονομάζεται αυτή η επιλογή κατανομής και την απώλεια , κατηγορηματικό αλγορίθμο (categorical algorithm) [11].

Algorithm 13: Categorical Algorithm

Input: A transition $s_t, a_t, r_t, s_{t+1}, \gamma \in [0, 1]$.
$Q(s_{t+1}, a) := \sum_i z_i p_i(s', a)$
$a* \leftarrow \arg\max_\alpha Q(s_{t+1}, a)$
for j=1 **to** N-1 **do**
  $\hat{\mathcal{T}}z_j \leftarrow [r_t + \gamma z_j]_{V_{min}}^{V_{max}}$
  $b_j \leftarrow (\hat{\mathcal{T}}z_j - V_{min})/\Delta z$
  $l \leftarrow \lfloor b_j \rfloor, u \leftarrow \lceil b_j \rceil$
  $m_l \leftarrow m_l + p_j(s_{t+1}, a^*)(u - b_j)$
  $m_u \leftarrow m_u + p_j(s_{t+1}, a^*)(b_j - l)$
end for
output $-\sum_i m_i \log p_i(s_t, a_t)$

3.2.7 Αλγόριθμος (Rainbow)

Κάθε ένας από αυτούς τους αλγόριθμους επιτρέπει ουσιαστική βελτίωση των επιδόσεων, μεμονωμένα. Εφόσον το κάνουν αυτό αντιμετωπίζοντας ριζικά διαφορετικά ζητήματα και επειδή βασίζονται σε ένα κοινό πλαίσιο, θα μπορούσαν εύλογα να συνδυαστούν σε έναν πράκτορα που συνδυάζει όλα τα προαναφερθέντα συστατικά. Αυτές οι διαφορετικές ιδέες μπορούν να ενσωματωθούν και είναι σε μεγάλο βαθμό συμπληρωματικές. Ενσωματώνουμε όλα τα προαναφερθέντα στοιχεία σε έναν ενιαίο ολοκληρωμένο πράκτορα, τον οποίο ονομάζουμε Rainbow [4].

Πρώτον, ξεκινάμε με την απώλεια του κατηγορηματικού αλγορίθμου (26) και αντικαθιστούμε το 1-βήμα με την παραλλαγή πολλαπλών βημάτων. Κατασκευάζουμε την κατανομή του στόχου αναθέτοντας για στόχο την κατανομή της αξίας του S_{t+n}:

$$d_t^{(n)} = (R_t^{(n)} + \gamma_t^{(n)}\boldsymbol{z},\ \boldsymbol{p}_{\bar{\theta}}(S_{t+n}, a_{t+n}^*)) \tag{27}$$

Η απώλεια που προκύπτει είναι

$$D_{KL}(\Phi_{\boldsymbol{z}} d_t^{(n)} || d_t) \tag{28}$$

όπου $\Phi_{\boldsymbol{z}}$ είναι η προβολή στο $\boldsymbol{z}$.

Συνδυάζουμε την απώλεια κατανομής πολλαπλών βημάτων με το double Q-Learning (14), χρησιμοποιώντας την άπληστη δράση στο S_{t+n} που επιλέγεται σύμφωνα με το *online* δίκτυο ως τη δράση bootstrap a^*_{t+n} και αξιολογούμε αυτήν δράση ($Q(a^*)$) χρησιμοποιώντας το δίκτυο στόχο.

Στην τυπική αναλογική παραλλαγή της επαναφοράς εμπειρίας με προτεραιότητα (κεφ. 3.2.4) χρησιμοποιείται το απόλυτο σφάλμα TD για να δοθεί προτεραιότητα στις μεταβάσεις. Αυτό μπορεί να υπολογιστεί χρησιμοποιώντας τις μέσες τιμές δράσης. Ωστόσο, μπορούμε να κάνουμε τον Rainbow να δώσει προτεραιότητα στις μεταβάσεις με βάση την απώλεια KL, αφού αυτήν ελαχιστοποιεί ο αλγόριθμος. Δηλαδή για την p_i στην (18) ισχύει:

$$p_t \propto D_{KL}(\Phi_{\boldsymbol{z}} d_t^{(n)} || d_t)^{\alpha} \tag{29}$$

Η αρχιτεκτονική του δικτύου είναι μια αρχιτεκτονική δικτύων με μονομαχίες (17) προσαρμοσμένη για χρήση με κατανομές επιστροφής. Το δίκτυο έχει μια κοινή αναπαράσταση $f_\xi(s)$, η οποία στη συνέχεια τροφοδοτείται σε μια ροή τιμών υ_η με N_{atoms} εξόδους,και σε ένα ρεύμα πλεονεκτημάτων a_ξ με $N_{atoms} \times N_{actions}$ εξόδους, όπου $a^i_\xi(f_\xi(s), a)$ θα υποδηλώνει την έξοδο που αντιστοιχεί στο άτομο i και τη δράση a. Για κάθε άτομο z^i, τα ρεύματα αξίας και πλεονεκτήματος συσσωματώνονται όπως και στο dueling DQN (17) και στη συνέχεια διέρχονται από ένα στρώμα softmax για να ληφθούν οι κανονικοποιημένες παραμετρικές κατανομές που χρησιμοποιούνται για την εκτίμηση των κατανομών των επιστροφών:

$$p^i_\theta(s, a) = \frac{\exp(\upsilon^i_\eta(\phi) + a^i_\psi(\phi, a) - \bar{a}^i_\psi(s))}{\sum_j \exp(\upsilon^j_\eta(\phi) + a^j_\psi(\phi, a) - \bar{a}^j_\psi(s))} \tag{30}$$

Όπου $\phi = f_\xi(s)$, και $\bar{a}^i_\psi$ ο μέσος των δράσεων a^i_ψ. Τέλος αντικαθιστούμε όλα τα γραμμικά επίπεδα με το θορυβώδες ισοδύναμό τους που περιγράφεται από την (19)

Επικεντρωθήκαμε εδώ σε μεθόδους που βασίζονται σε συναρτήσεις αξίας, στην οικογένεια Q-learning. Στην επόμενη ενότητα εξετάζουμε μια διαφορετική οικογένια αλγορίθμωμ RL με βάση την πολιτική.

3.3 Μέθοδοι κλίσης πολιτικής(Policy Gradient Methods)

3.3.1 Προσέγγιση της Πολιτικής και το Θεώρημα Κλίσης Πολιτικής (policy gradient theorem)

Σε αυτό το κεφάλαιο θεωρούμε κάτι νέο. Μέχρι στιγμής εξετάσαμε μόνο μεθόδους που μαθαίνουν τις αξίες των δράσεων και στη συνέχεια επιλέγουν δράσεις βάσει των εκτιμώμενων αξιών δράσης τους. Οι πολιτικές τους δεν θα υπήρχαν

καν χωρίς τις εκτιμήσεις της αξίας-δράσης. Αυτή τη φορά εξετάζουμε μεθόδους που μαθαίνουν μια *παραμετροποιημένη πολιτική* που μπορεί να επιλέξει ενέργειες χωρίς να συμβουλευτεί μια συνάρτηση τιμής. Μια συνάρτηση τιμής μπορεί να χρησιμοποιηθεί για να μάθει την παραμετροποιημένη πολιτική, αλλά δεν απαιτείται για την επιλογή ενέργειας.

Χρησιμοποιούμε το συμβολισμό $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ για το διάνυσμα παραμέτρων της πολιτικής. Οπότε γράφουμε $\pi(\alpha|s, \boldsymbol{\theta}) = \Pr\{A_t = \alpha | S_t = s, \boldsymbol{\theta_t} = \boldsymbol{\theta}\}$ για την πιθανότητα ότι η δράση α λαμβάνεται στο χρόνο t δεδομένου ότι το περιβάλλον είναι σε κατάσταση s κατά το χρόνο t με την παράμετρο $\boldsymbol{\theta}$. Όπως θα δούμε αργότερα, είναι πιθανό μια τέτοια μέθοδος να χρησιμοποιεί επίσης μια συνάρτηση τιμών με βοηθητικό τρόπο για να αυξήσει την αποτελεσματικότητα της πολιτικής. Σε αυτή την περίπτωση τότε το διάνυσμα βάρους της συνάρτησης τιμών συμβολίζεται $\mathbf{w} \in \mathbb{R}^d$, όπως $\hat{v}(s, \mathbf{w})$.

Αναφέραμε στο κεφάλαιο 2 τις μεθόδους κλίσης και τον τρόπο με τον οποίο χρησιμοποιούνται ως βάση για την οπισθοδιάδοση των νευρωνικών δικτύων. Θα τις επανεξετάσουμε εδώ, ανεξάρτητα από το αν η πολιτική διαμορφώνεται με ένα νευρωνικό δίκτυο (αν και θα χρησιμοποιήσουμε ένα τέτοιο μοντέλο αργότερα). Συγκεκριμένα, χρησιμοποιούμε την ***κλίση** ανόδου (gradient ascent)*, δηλαδή προσπαθούμε να μεγιστοποιήσουμε ένα μέτρο απόδοσης $J(\theta)$ σε σχέση με την παράμετρο πολιτικής:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} + \alpha \widehat{\nabla J(\boldsymbol{\theta_t})} \tag{31}$$

όπου $\widehat{\nabla J(\boldsymbol{\theta_t})}$ είναι μια στοχαστική εκτίμηση της οποίας η αναμενόμενη τιμή προσεγγίζει την κλίση του μέτρου απόδοσης σε σχέση με το όρισμά της $\boldsymbol{\theta_t}$.

Όλες οι μέθοδοι που ακολουθούν αυτό το γενικό σχήμα ονομάζονται *μέθοδοι κλίσης πολιτικής*, ανεξάρτητα από το αν μαθαίνουν επίσης μια συνάρτηση αξίας. Οι μέθοδοι που μαθαίνουν τις προσεγγίσεις τόσο στις πολιτικές όσο και τις συναρτήσεις αξίας ονομάζονται συχνά μέθοδοι δράστη-κριτή(actor-critic), όπου ο «δράστης» είναι μια αναφορά στην πολιτική που μαθαίνεται και ο «κριτής» αναφέρεται στη συνάρτηση της τιμής που μαθαίνεται, συνήθως μια συνάρτηση τιμής-κατάστασης.

Σε μεθόδους κλίσης πολιτικής, η πολιτική μπορεί να παραμετροποιηθεί με οποιοδήποτε τρόπο, αρκεί η $\pi(a|s, \boldsymbol{\theta})$ να είναι διαφορίσιμη σε σχέση με τις παραμέτρους της, δηλαδή, εφόσον υπάρχει $\nabla_{\boldsymbol{\theta}} \pi(a|s, \boldsymbol{\theta})$ και είναι πεπερασμένο. Στην πράξη, για να εξασφαλίσουμε την εξερεύνηση, απαιτούμε γενικά η πολιτική να μην γίνεται ποτέ ντετερμινιστική (δηλ. Ότι $\pi(a|s, \boldsymbol{\theta}) \in (0, 1), \forall s, a, \boldsymbol{\theta}$). Αν και πολλά μοντέλα μπορούν και έχουν χρησιμοποιηθεί, σε αυτήν την έργασία χρησιμοποιούμε νευρωνικά δίκτυα συγκρίσιμα με τα DQN που συζητήσαμε προηγουμένως.

Εκτός από τα πρακτικά πλεονεκτήματα της παραμετροποίησης της πολιτικής σε σύγκριση με τις ϵ-άπληστες μεθόδους, υπάρχει επίσης ένα σημαντικό θεωρητικό πλεονέκτημα. Με τη συνεχή παραμετροποίηση πολιτικής, οι πιθανότητες δράσης αλλάζουν ομαλά ως συνάρτηση των παραμέτρων, ενώ στην ϵ-άπληστη επιλογή οι πιθανότητες δράσης μπορεί να αλλάξουν δραματικά για μια αυθαίρετα μικρή αλλαγή στις εκτιμώμενες τιμές δράσης, εάν η αλλαγή αυτή έχει ως αποτέλεσμα μια διαφορετική ενέργεια που έχει μέγιστη τιμή. Εξαιτίας αυτής της ισχυρότερης σύγκλισης, υπάρχουν διαθέσιμες εγγυήσεις για τις μεθόδους κλίσης πολιτικής από

ό,τι για τις μεθόδους δράσης-αξίας. Συγκεκριμένα, η συνέχιση της εξάρτησης της πολιτικής από τις παραμέτρους που επιτρέπει στις μεθόδους βαθμίδωσης πολιτικής να προσεγγίσουν την άνοδο της κλίσης (31).

Οι επεισοδιακές και συνεχείς περιπτώσεις ορίζουν το μέτρο απόδοσης $J(\boldsymbol{\theta})$, διαφορετικά και έτσι πρέπει να αντιμετωπιστούν ξεχωριστά σε κάποιο βαθμό. Παρ΄όλα αυτά, θα προσπαθήσουμε να παρουσιάσουμε ομοιόμορφα και τις δύο περιπτώσεις και θα αναπτύξουμε ένα συμβολισμό έτσι ώστε τα μεγάλα θεωρητικά αποτελέσματα να μπορούν να περιγραφούν με ένα ενιαίο σύνολο εξισώσεων.

Σε αυτή την ενότητα αντιμετωπίζουμε την επεισοδιακή περίπτωση, για την οποία ορίζουμε το μέτρο απόδοσης ως την τιμή της αρχικής κατάστασης του επεισοδίου. Ορίζουμε την απόδοση ως:

$$J(\boldsymbol{\theta}) \doteq v_{\pi_\theta}(s_0) \tag{32}$$

Με την προσέγγιση συναρτήσεων, φαίνεται δύσκολο να αλλάξουμε την παράμετρο της πολιτικής με τρόπο που να εξασφαλίζει βελτίωση. Το πρόβλημα είναι ότι η απόδοση εξαρτάται τόσο από τις επιλογές δράσης όσο και από την κατανομή των καταστάσεων στις οποίες πραγματοποιούνται αυτές οι επιλογές και ότι και οι δύο επηρεάζονται από την παράμετρο πολιτικής. Με δεδομένη μία κατάσταση, το αποτέλεσμα της πολιτικής παραμέτρων στις ενέργειες, και συνεπώς στην ανταμοιβή,μπορεί να υπολογιστεί με σχετικά απλό τρόπο από τη γνώση της παραμετροποίησης. Αλλά η επίδραση της πολιτικής στην κατανομή των καταστάσεων είναι συνάρτηση του περιβάλλοντος και είναι συνήθως άγνωστη. Πώς μπορούμε να υπολογίσουμε την κλίση απόδοσης σε σχέση με την παράμετρο της πολιτικής όταν η κλίση εξαρτάται από το άγνωστο αποτέλεσμα αλλαγών πολιτικής στην κατανομή κατάστασης.

Ευτυχώς, υπάρχει μια εξαιρετική θεωρητική απάντηση σε αυτή την πρόκληση με τη μορφή του θεώρηματος κλίσης πολιτικής, το οποίο μας παρέχει μια αναλυτική έκφραση για την κλίση της απόδοσης σε σχέση με την παράμετρο πολιτικής (που είναι αυτό που πρέπει να προσεγγίσουμε για την άνοδο κλίσης (31) και δεν περιλαμβάνει την παράγωγο της κατανομής των καταστάσεων).

Theorem : Policy Gradient Theorem

Το θεώρημα κλισης πολιτικής δηλώνει ότι το μέτρο απόδοσης είναι ανάλογο με τις κλίσεις της πολιτικής:

$$\nabla J(\boldsymbol{\theta}) = \nabla v_\pi(s_0) \propto \mu(s) \sum_\alpha q_\pi(s,a) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta}) \tag{33}$$

3.3.2 REINFORCE: Monte Carlo Policy Gradient

Είμαστε πλέον έτοιμοι για τον πρώτο μας αλγόριθμο μάθησης με βάση την πολιτική. Η γενική στρατηγική μας για την στοχαστική άνοδο κλίσης απαιτεί έναν τρόπο λήψης δειγμάτων έτσι ώστε η προσδοκία της κλίσης του δείγματος να είναι ανάλογη με την πραγματική κλίση του μέτρου απόδοσης ως συνάρτηση της παραμέτρου. Οι κλίσεις των δειγμάτων πρέπει να είναι ανάλογες με την κλίση, επειδή οποιαδήποτε σταθερά αναλογικότητας μπορεί να απορροφηθεί στο μέγεθος βήματος α, το οποίο είναι διαφορετικά αυθαίρετο. Το θεώρημα κλίσης πολιτικής δίνει μια ακριβή έκφραση ανάλογη με την κλίση. το μόνο που χρειάζεται είναι κάποιος τρόπος δειγματοληψίας, του οποίου η προσδοκία ισούται ή προσεγγίζει αυτή την έκφραση. Παρατηρήστε ότι η δεξιά πλευρά του θεώρημα της κλίσης πολιτικής είναι ένα άθροισμα πάνω από τις καταστάσεις που σταθμίζονται από το πόσο συχνά συμβαίνουν σύμφωνα με την πολιτική στόχου π· αν το π ακολουθείται, τότε οι καταστάσεις θα συναντηθούν σε αυτές τις αναλογίες. Έτσι:

$$\nabla J(\boldsymbol{\theta}) = \nabla v_\pi(s_0) \propto \mu(s) \sum_\alpha \nabla \pi(a) q_\pi(s,a) \tag{34}$$

$$= \mathbb{E}\left[\sum_\alpha q_\pi(S_t,\alpha)\nabla_\theta \pi(\alpha|S_t,\boldsymbol{\theta})\right] \tag{35}$$

Θα θέλαμε να το μεταφέρουμε περαιτέρω και να χειριστούμε την ενέργεια με τον ίδιο τρόπο (αντικαθιστώντας a με τη δράση δείγματος A_t). Το υπόλοιπο μέρος της παραπάνω αναμενόμενης τιμής είναι ένα άθροισμα στο σύνολο των δράσεων. Αν σταθμιστεί κάθε όρος από την πιθανότητα επιλογής των δράσεων, δηλαδή, σύμφωνα με το $\pi(a|S_t,\boldsymbol{\theta})$, τότε η αντικατάσταση θα μπορούσε να γίνει. Μπορούμε να το πραγματοποιήσουμε πολλαπλασιάζοντας και διαιρώντας με αυτή την πιθανότητα. Συνεχίζοντας από την προηγούμενη εξίσωση, αυτό δίνει:

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_\pi\left[\sum_\alpha \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,\alpha)\frac{\nabla_{\boldsymbol{\theta}}\pi(\alpha|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})}\right] \tag{36}$$

$$= \mathbb{E}_\pi\left[q_\pi(S_t,\alpha)\frac{\nabla_{\boldsymbol{\theta}}\pi(\alpha|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})}\right] \tag{37}$$

$$= \mathbb{E}_\pi\left[G_t\frac{\nabla_{\boldsymbol{\theta}}\pi(\alpha|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})}\right] \tag{38}$$

όπου G_t είναι η επιστροφή ως συνήθως. Η τελική έκφραση στις παρενθέσεις είναι ακριβώς αυτό που χρειάζεται, μια ποσότητα που μπορεί να δειγματοληφθεί σε κάθε βήμα του οποίου η προσδοκία είναι ίση με την κλίση. Χρησιμοποιώντας αυτό το δείγμα για να δημιουργήσουμε τον γενικό αλγόριθμο ανόδου στοχαστικής κλίσης, μας αποδίδει την ενημέρωση:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha G_t \frac{\nabla_{\boldsymbol{\theta}}\pi(\alpha|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})} \tag{39}$$

Ονομάζουμε αυτόν τον αλγόριθμο REINFORCE. Η ενημέρωσή του έχει μια διαισθητική εξήγηση. Κάθε αύξηση είναι ανάλογη με το προϊόν μιας επιστροφής G_t και ενός διανύσματος, η κλίση της πιθανότητας λήψης της ενέργειας που πράγματι λαμβάνεται, διαιρεμένη με την πιθανότητα ανάληψης αυτής της ενέργειας. Το διάνυσμα είναι η κατεύθυνση στο χώρο των παραμέτρων που αυξάνει περισσότερο την πιθανότητα επανάληψης της ενέργειας A_t στις μελλοντικές επισκέψεις στην κατάσταση S_t. Η ενημέρωση αυξάνει το διάνυσμα παραμέτρων προς αυτήν την κατεύθυνση ανάλογα προς την απόδοση και αντιστρόφως ανάλογα προς την πιθανότητα δράσης. Το πρώτο έχει νόημα επειδή προκαλεί την κλίση να κινείται περισσότερο στις κατευθύνσεις που ευνοούν τις ενέργειες που αποφέρουν την υψηλότερη απόδοση. Το τελευταίο έχει νόημα γιατί διαφορετικά οι ενέργειες που επιλέγονται συχνά είναι σε πλεονέκτημα (οι ενημερώσεις θα είναι συχνότερα προς την κατεύθυνσή τους) και μπορεί να κερδίσουν ακόμα και αν δεν αποφέρουν την υψηλότερη απόδοση.

Σημειώστε ότι ο REINFORCE χρησιμοποιεί την πλήρη απόδοση από την στιγμή t, η οποία περιλαμβάνει όλες τις μελλοντικές ανταμοιβές μέχρι το τέλος του επεισοδίου. Υπό αυτή την έννοια, ο REINFORCE είναι ένας αλγόριθμος Monte Carlo και είναι σαφώς καθορισμένος μόνο για την επεισοδιακή περίπτωση με όλες τις ενημερώσεις να γίνονται μετά την ολοκλήρωση του επεισοδίου.

Algorithm 14: REINFORCE,Monte-Carlo Policy-Gradient (episodic)

Input: a differentiable policy parameterization $\pi(\alpha|s, \boldsymbol{\theta})$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$
Repeat forever:
  Generate an episode $S_0, A_0, R_1, ..., S_{T-1}, A_{T-1}, R_T$ following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  For each step of the episode $t = 0, 1, 2, ..., T-1$:
  $G \leftarrow$ return from step t
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + a\gamma^t G \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t, \boldsymbol{\theta})$

3.3.3 REINFORCE με μέτρο σύγκρισης (baseline)

Το θεώρημα κλίσης πολιτικής (33) μπορεί να γενικευτεί ώστε να περιλαμβάνει σύγκριση της τιμής δράσης με ένα αυθαίρετο μέτρο σύγκρισης b(s):

$$\nabla J(\boldsymbol{\theta}) = \nabla v_\pi(s_0) \propto \mu(s) \sum_\alpha (q_\pi(s,a) - b(s)) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta}) \tag{40}$$

Το μέτρο σύγκρισης μπορεί να είναι οποιαδήποτε συνάρτηση, ακόμη και μια τυχαία μεταβλητή, εφ ΄όσον δεν μεταβάλλεται με a. Η εξίσωση παραμένει έγκυρη επειδή η αφαιρεθείσα ποσότητα είναι μηδέν:

$$\sum_a b(s) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta}) = b(s) \nabla_{\boldsymbol{\theta}} \sum_a \pi(a|s,\boldsymbol{\theta}) = b(s) \nabla_{\boldsymbol{\theta}} 1 = 0$$

Το θεώρημα κλίσης πολιτικής με βάση μπορεί να χρησιμοποιηθεί για την εξαγωγή ενός κανόνα ενημέρωσης χρησιμοποιώντας παρόμοια βήματα όπως στην προηγούμενη ενότητα. Ο κανόνας ενημέρωσης που καταλήγουμε είναι μια νέα έκδοση του REINFORCE που περιλαμβάνει μια γενική γραμμή βάσης:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha (G_t - b(S_t)) \frac{\nabla_{\boldsymbol{\theta}} \pi(\alpha|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})} \tag{41}$$

Επειδή το μέτρο σύγκρισης μπορεί να είναι ομοιόμορφα μηδενικό, αυτή η ενημέρωση είναι μια αυστηρή γενίκευση του REINFORCE. Γενικά, το μέτρο σύγκρισης αφήνει αμετάβλητη την αναμενόμενη αξία της ενημέρωσης, αλλά μπορεί να έχει μεγάλο αποτέλεσμα σχετικά με τη διακύμανσή του. Μια φυσική επιλογή για το μέτρο σύγκρισης είναι μια εκτίμηση της τιμής κατάστασης, $\hat{v}(S_t, \mathbf{w})$, όπου $\mathbf{w} \in \mathbb{R}^m$ είναι ένα διάνυσμα βάρους που μαθαίνεται με μία από τις μεθόδους που παρουσιάστηκαν στα προηγούμενα κεφάλαια. Επειδή το REINFORCE είναι μια μέθοδος Monte Carlo για την εκμάθηση της παραμέτρου πολιτικής, θ, φαίνεται φυσικό να χρησιμοποιήσουμε επίσης μια μέθοδο Monte Carlo για να μάθουμε τα βάρη της κατάστασης αξίας, $\mathbf{w}$.

Ένας πλήρης αλγόριθμος ψευδοκώδικα για το REINFORCE με μέτρο σύγκρισης δίδεται εδώ χρησιμοποιώντας μια τέτοια συνάρτηση κατάστασης-τιμής ,που μαθαίνεται, ως μέτρο σύγκρισης. Αυτός ο αλγόριθμος έχει δύο ρυθμούς εκμάθησης, που δηλώνουν $\alpha_{\boldsymbol{\theta}}$ και $\alpha_{\mathbf{w}}$. Το μέγεθος βήματος για την συνάρτηση κατάσταση-τιμής (εδώ $\alpha_{\mathbf{w}}$) είναι σχετικά εύκολο. Είναι πολύ λιγότερο σαφές πώς να ορίσετε το μέγεθος βήματος $\alpha_{\boldsymbol{\theta}}$ για τις παραμέτρους πολιτικής. Εξαρτάται από το φάσμα των διακυμάνσεων των ανταμοιβών και από την παραμετροποίηση της πολιτικής.

Algorithm 15: REINFORCE,Monte-Carlo Policy-Gradient (episodic)

Input: a differentiable policy parameterization $\pi(\alpha|s, \boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s, \mathbf{w})$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$, and state-value weights $\mathbf{w} \in \mathbb{R}^d$
Repeat forever:
  Generate an episode $S_0, A_0, R_1, ..., S_{T-1}, A_{T-1}, R_T$ following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  For each step of the episode $t = 0, 1, 2, ..., T-1$:
  $G \leftarrow$ return from step t
  $\delta \leftarrow G_t - \hat{v}(S_t, \mathbf{w})$
  $\mathbf{w} \leftarrow w + \alpha_{\mathbf{w}} \gamma^t \delta \nabla_{\mathbf{w}} \hat{v}(S_t, \mathbf{w})$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \gamma^t \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t, \boldsymbol{\theta})$

3.3.4 Μέθοδοι Δράστη-Κριτή (Actor-Critic)

Παρόλο που η μέθοδος REINFORCE με μέτρο σύγκρισης μαθαίνει τόσο την πολιτική όσο και τη συνάρτηση της τιμής κατάστασης, δεν θεωρούμε ότι είναι μια μέθοδος κριτή-δράστη, επειδή η συνάρτηση της κατάστασης τιμής χρησιμοποιείται μόνο ως γραμμή βάσης και όχι ως κριτής. Δηλαδή, δεν χρησιμοποιείται για bootstrapping (ενημέρωση της εκτίμησης αξίας για μια κατάσταση από τις εκτιμώμενες τιμές των επόμενων καταστάσεων), αλλά μόνο ως βάση για την κατάσταση της οποίας η εκτίμηση ενημερώνεται. Αυτή είναι μια χρήσιμη διάκριση, διότι μόνο με την τεχνική bootstrap εισάγουμε προκατάληψη και ασυμπτωτική εξάρτηση από την ποιότητα της προσέγγισης της συνάρτησης.

Η προκατάληψη που εισάγεται με (bootstrapping) και την εξάρτηση από την εκπροσώπηση της κατάστασης είναι συχνά επωφελής επειδή μειώνει τη διακύμανση και επιταχύνει την εκμάθηση. Το REINFORCE με το μέτρο σύγκρισης είναι αμερόληπτο και θα συγκλίνει ασυμπτωτικά σε ένα τοπικό ελάχιστο, αλλά όπως όλες οι μέθοδοι του Monte Carlo τείνει να μαθαίνει αργά (να παράγει εκτιμήσεις υψηλής διακύμανσης) και να είναι ακατάλληλο για την εφαρμογή online ή για συνεχή προβλήματα.

Όπως είδαμε νωρίτερα με μεθόδους χρονικής διαφοράς, μπορούμε να εξαλείψουμε αυτές τις δυσκολίες και μέσω μεθόδων πολλαπλών βημάτων μπορούμε να επιλέξουμε με ευελιξία το βαθμό του bootstrapping. Προκειμένου να αποκτήσουμε αυτά τα πλεονεκτήματα στην περίπτωση μεθόδων κλίσης πολιτικής, χρησιμοποιούμε μεθόδους δράστη-κριτή και χρησιμοποιούμε τον κριτή για bootstrap:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha(G_{t:t+1} - \hat{v}(S_t), \mathbf{w}) \frac{\nabla_{\boldsymbol{\theta}} \pi(\alpha|S_t, \boldsymbol{\theta})}{\pi(a|S_t, \boldsymbol{\theta})} \tag{42}$$

$$= \boldsymbol{\theta}_t + \alpha(R_{t+1} + \gamma \hat{v}(S_{t_1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w})) \frac{\nabla_{\boldsymbol{\theta}} \pi(\alpha|S_t, \boldsymbol{\theta})}{\pi(a|S_t, \boldsymbol{\theta})} \tag{43}$$

$$= \boldsymbol{\theta}_t + \alpha \delta_t \frac{\nabla_{\boldsymbol{\theta}} \pi(\alpha|S_t, \boldsymbol{\theta})}{\pi(a|S_t, \boldsymbol{\theta})} \tag{44}$$

Σημειώστε ότι τώρα είναι ένας πλήρως online, αυξητικός αλγόριθμος, με τις

καταστάσεις, τις ενέργειες και τις ανταμοιβές να επεξεργάζονται καθώς εμφανίζονται και στη συνέχεια δεν επανεξετάζονται ποτέ.

Algorithm 16: One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(\alpha|s, \boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s, \mathbf{w})$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$, and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever:
  Initialize S (first state of episode)
  $I \leftarrow 1$
  While S is not terminal:
    Take action $A \sim \pi(\cdot|S, \boldsymbol{\theta})$, observe S', R
    $\delta \leftarrow R_{t+1} + \gamma \hat{v}(S_{t_1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w})$
    $\mathbf{w} \leftarrow w + \alpha_{\mathbf{w}} \gamma^t I \delta \nabla_{\mathbf{w}} \hat{v}(S_t, \mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \gamma^t I \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t, \boldsymbol{\theta})$
    $I \leftarrow \gamma I, S \leftarrow S'$

3.3.5 Εγγύς Βελτιστοποίηση Πολιτικής (Proximal Policy Optimization-PPO)

Μέχρι στιγμής συζητήσαμε μεθόδους κλίσης πολιτικής που λειτουργούν υπολογίζοντας έναν εκτιμητή της κλίσης της πολιτικής και την ενσωμάτωσή του σε αλγόριθμο ανόδου στοχαστικής κλίσης. Η αντικειμενική συνάρτηση της οποίας η κλίση πρέπει να εκτιμηθεί είναι:

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t|s_t) \hat{A}_t \right] \tag{45}$$

Ενώ είναι ελκυστικό να εκτελεστούν πολλαπλά βήματα βελτιστοποίησης σε αυτή την απώλεια L^{PG} χρησιμοποιώντας την ίδια τροχιά, αυτό δεν είναι καλά αιτιολογημένο, και εμπειρικά συχνά οδηγεί σε καταστροφικά μεγάλες ενημερώσεις πολιτικής. Ως εναλλακτική λύση, μεγιστοποιούμε μια αντιπρόσωπη αντικειμενική συνάρτηση. Συγκεκριμένα, κόβουμε την αναλογία μεταξύ της νέας και της παλαιάς πολιτικής. Έστω $r_t(\theta)$ ο λόγος των πιθανοτήτων $r_t(\theta) = \frac{\pi(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$. Η αντιπρόσωπη αντικειμενική συνάρτηση $L^{CPI}(\theta) = \hat{\mathbb{E}}_t \left[r_t(\theta) \hat{A}_t \right]$. Χρησιμοποιούμε αυτό ως βάση για τη νέα αντικειμενική συνάρτηση:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right] \tag{46}$$

όπου ϵ μια υπερπαράμετρος, ας πούμε 0.2. Το κίνητρο για αυτό το στόχο έχει ως εξής. Ο πρώτος όρος μέσα στο min είναι L^{CPI}. Ο δεύτερος όρος, $clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)$, τροποποιεί την εν λόγω αντιπρόσωπη αντικειμενική συνάρτηση αποκόπτοντας τον λόγο πιθανοτήτων. Αφαιρεί έτσι τη δυνατότητα για τη μετακίνηση του r_t εκτός του διαστήματος $[1-\epsilon, 1+\epsilon]$. Τέλος, λαμβάνουμε το ελάχιστο του

αποκομμένου και μη στόχου, οπότε ο τελικός στόχος είναι ένα κατώτερο όριο (δηλ. ένα απαισιόδοξο όριο) στον αποκομμένο στόχο.

Οι υποκατάστατες απώλειες από τα προηγούμενα τμήματα μπορούν να υπολογιστούν και να διαφοροποιηθούν με μια μικρή αλλαγή σε μια τυπική εφαρμογή κλίσης πολιτικής. Ένα στυλ εφαρμογής πολιτικής κλίσης, εκτελεί την πολιτική για τα χρονικά σημεία T (όπου το T είναι πολύ μικρότερο από το μήκος του επεισοδίου) και χρησιμοποιεί τα συλλεχθέντα δείγματα για μια ενημέρωση. Αυτό το στυλ απαιτεί έναν εκτιμητή πλεονεκτήματος ο οποίος δεν κοιτάζει πέρα από το χρονοδιάγραμμα T:

$$\hat{A}_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + ... + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T) - V(s_t) \quad (47)$$

όπου t ορίζει το δείκτη χρόνου στο $[0, T]$. Ένας αλγόριθμος εγγύς βελτιστοποίηση πολιτικής (PPO) που χρησιμοποιεί τμήματα τροχιάς σταθερού μήκους φαίνεται παρακάτω. Σε κάθε επανάληψη, καθένας από τους N (παράλληλους) δράστες συλλέγει τα T δεδομένα. Στη συνέχεια, κατασκευάζουμε την υποκατασταθείσα απώλεια σε αυτές τις NT χρονοσειρές δεδομένων και την βελτιστοποιούμε με το minibatch SGD.

Algorithm 17: PPO, Actor-Critic Style

for iteration = 1,2,... **do**
  for actor = 1,2,...,N **do**
    Run policy $\pi_{\theta_{old}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, \hat{A}_2, ..., \hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and
  minibatch size $M \leq NT$
  $\theta_{old} \leftarrow \theta$
end for

Αφού ολοκληρώσαμε τη θεωρία πίσω από τους χρησιμοποιούμενους αλγορίθμους, προχωρούμε τώρα στα πειράματα, εφαρμόζοντας και συγκρίνοντας τους δύο αλγόριθμους που αποτελούσαν, στη συγκεκριμένη εργασία, το αποκορύφωμα της κάθε μιας από τις δύο οικογένειες αλγορίθμων, δηλαδή τον Rainbow και τον PPO.

4 Υλοποίηση και Αξιολόγηση Πειραμάτων

4.1 Περιγραφή Περιβάλλοντος

4.1.1 Εισαγωγή

Αρχικά θα παρουσιάσουμε το benchmark που χρησιμοποιήθηκε, ένα περιβάλλον βασισμένο στη σειρά βιντεοπαιχνιδιών Sonic the Hedgehog™. Αυτό το benchmark, προορίζεται για τη μέτρηση της απόδοσης των αλγορίθμων μεταφοράς μάθησης στον τομέα RL.

4.1.2 Gym Retro

Το Sonic benchmark υπόκειται στο Gym Retro, ένα έργο που στοχεύει στη δημιουργία περιβάλλοντος RL από διάφορα προσομοιωμένα βιντεοπαιχνίδια. Στον πυρήνα του Gym Retro βρίσκεται το πακέτο gym-retro Python, το οποίο εκθέτει προσομοιωμένα παιχνίδια ως περιβάλλοντα Gym. Όπως το RLE, το gym-retro χρησιμοποιεί το libretro API για διασύνδεση με εξομοιωτές παιχνιδιών, καθιστώντας πολύ εύκολο να προσθέσετε νέους εξομοιωτές στο gym-retro.

Το πακέτο gym-retro περιλαμβάνει ένα σύνολο δεδομένων παιχνιδιών. Κάθε παιχνίδι στο σύνολο δεδομένων αποτελείται από ROM, μία ή περισσότερες καταστάσεις αποθήκευσης, ένα ή περισσότερα σενάρια και ένα αρχείο δεδομένων. Ακολουθούν περιγραφές υψηλού επιπέδου για καθένα από αυτά τα στοιχεία:

- ROM - τα δεδομένα και ο κώδικας που αποτελούν ένα παιχνίδι που φορτώθηκε από τον εξομοιωτή για να παίξει αυτό παιχνίδι.
- Save state - ένα στιγμιότυπο της κατάστασης της κονσόλας σε κάποιο σημείο του παιχνιδιού. Για παράδειγμα, θα μπορούσε να δημιουργηθεί μια κατάσταση αποθήκευσης για την αρχή κάθε επιπέδου.
- Data File - ένα αρχείο που περιγράφει πού αποθηκεύονται διάφορα κομμάτια πληροφοριών στην μνήμη της κονσόλας. Για παράδειγμα, ένα αρχείο δεδομένων μπορεί να υποδεικνύει πού βρίσκεται η βαθμολογία.
- Scenario - περιγραφή των συνθηκών και των συναρτήσεων ανταμοιβής. Ένα αρχείο σεναρίου μπορεί να αναφέρει πεδία από το αρχείο δεδομένων.

4.1.3 The Sonic Video Game

Σχήμα 4: Στιγμιότυπα οθόνης από Sonic 3 Knuckles. Αριστερά: μια κατάσταση όπου ο παίκτης μπορεί να πηδήξει στον αέρα χρησιμοποιώντας ένα αντικείμενο με δυναμική τραμπάλας (Mushroom Hill Zone, Act 2). Δεξιά: μια πόρτα που ανοίγει όταν ο παίκτης πηδά πάνω σε ένα κουμπί (Hydro City Zone, Act 1). Κάτω: μια κούνια όπου ο παίκτης πρέπει να πηδήξει την κατάλληλη στιγμή για να φτάσει σε μια υψηλή πλατφόρμα (Mushroom Hill Act 2)

Σε αυτό το benchmark, χρησιμοποιούμε τρία παρόμοια παιχνίδια: Sonic The Hedgehog™, Sonic The Hedgehog ™2, Sonic 3 Knuckles. Όλα αυτά τα παιχνίδια έχουν πολύ παρόμοιους κανόνες και τρόπους ελέγχου, αν και υπάρχουν λεπτές διαφορές μεταξύ τους (π.χ. Sonic 3 Knuckles περιλαμβάνει μερικά επιπλέον στοιχεία ελέγχου και χαρακτήρες). Χρησιμοποιούμε πολλά παιχνίδια για να λάβουμε όσο το δυνατόν περισσότερα περιβάλλοντα για το σύνολο δεδομένων μας.

Κάθε παιχνίδι Sonic χωρίζεται σε ζώνες (zone) και κάθε ζώνη χωρίζεται περαιτέρω σε πράξεις (act). Ενώ οι κανόνες και ο γενικός στόχος παραμένουν οι ίδιοι σε ολόκληρο το παιχνίδι, κάθε ζώνη έχει ένα μοναδικό σύνολο γραφικών και αντικειμένων. Διαφορετικές πράξεις σε μια ζώνη τείνουν να μοιράζονται αυτές τα γραφικά και αντικείμενα, αλλά διαφέρουν στη χωρική διάταξη. Θα αναφερθούμε σε μια (ROM, zone, act) πλειάδα ως «επίπεδο» (level).

Τα παιχνίδια Sonic παρέχουν ένα πλούσιο σύνολο προκλήσεων για τον παίκτη. Για παράδειγμα, ορισμένες ζώνες περιλαμβάνουν πλατφόρμες στις οποίες πρέπει να μεταβεί η συσκευή αναπαραγωγής για να ανοίξει πόρτες. Άλλες ζώνες απαιτούν από τον παίκτη να πηδήξει πρώτα σε έναν μοχλό για να στείλει ένα βλήμα

στον αέρα και μετά να περιμένει το βλήμα να πέσει πίσω στο μοχλό για να στείλει τη συσκευή αναπαραγωγής πάνω από κάποιο εμπόδιο. Μία ζώνη έχει ακόμη μια κούνια από την οποία ο παίκτης πρέπει να ξεφύγει σε μια ακριβή στιγμή για να ξεκινήσει το Sonic σε μια υψηλότερη πλατφόρμα. Παραδείγματα αυτών των προκλήσεων παρουσιάζονται στο Σχήμα 1.

4.1.4 Τα επίπεδα

Το benchmark αποτελείται από συνολικά 58 καταστάσεις αποθήκευσης που λαμβάνονται από τρία διαφορετικά παιχνίδια, όπου καθεμία από αυτές τις καταστάσεις αποθήκευσης έχει τον παίκτη στην αρχή ενός διαφορετικού επιπέδου. Ορισμένες πράξεις από τα αρχικά παιχνίδια δεν χρησιμοποιήθηκαν επειδή περιείχαν μόνο μάχες ενάντια σε αφεντικό ή επειδή δεν ήταν συμβατές με τη συνάρτηση ανταμοιβής μας.

Διαχωρίσαμε το σετ δοκιμής (test set) επιλέγοντας τυχαία ζώνες με περισσότερες από μία πράξεις και στη συνέχεια επιλέγοντας τυχαία μια πράξη από κάθε επιλεγμένη ζώνη. Σε αυτήν τη ρύθμιση, το σύνολο δοκιμών περιέχει κυρίως αντικείμενα και γραφικά που υπάρχουν στο εκπαιδευτικό σετ, αλλά με διαφορετικές διατάξεις.

Τα επίπεδα δοκιμής παρατίθενται στον παρακάτω πίνακα:

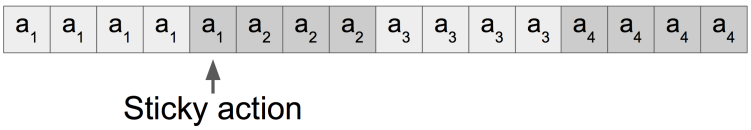
SonicTheHedgehog-Genesis	SpringYardZone.Act1
SonicTheHedgehog-Genesis	GreenHillZone.Act2
SonicTheHedgehog-Genesis	StarLightZone.Act3
SonicTheHedgehog-Genesis	ScrapBrainZone.Act1
SonicTheHedgehog2-Genesis	MetropolisZone.Act3
SonicTheHedgehog2-Genesis	HillTopZone.Act2
SonicTheHedgehog2-Genesis	CasinoNightZone.Act2
SonicAndKnuckles3-Genesis	LavaReefZone.Act1
SonicAndKnuckles3-Genesis	FlyingBatteryZone.Act2
SonicAndKnuckles3-Genesis	HydrocityZone.Act1
SonicAndKnuckles3-Genesis	AngelIslandZone.Act2

4.1.5 Frame Skip

Όλα τα περιβάλλοντα στη ενισχυτική μάθηση υλοποιούνται με μια μέθοδο step() που την προχωρεί κατά ένα βήμα.Η μέθοδος step() σε ακατέργαστα περιβάλλοντα gym-retro προωθεί το παιχνίδι κατά περίπου $\frac{1}{60}$ του δευτερολέπτου. Ωστόσο, ακολουθώντας την κοινή πρακτική για ALE (Arcade Learning Environment)[12] περιβάλλοντα, απαιτούμε τη χρήση ενός πλαισίου παράλειψης 4. Έτσι, από εδώ και πέρα, θα χρησιμοποιήσουμε τα χρονικά βήματα ως την κύρια μονάδα μέτρησης του χρόνου στο παιχνίδι. Με παράλειψη καρέ 4, ένα χρονικό βήμα αντιπροσωπεύει περίπου το $\frac{1}{15}$ του δευτερολέπτου.

Επιπλέον, δεδομένου ότι τα ντετερμινιστικά περιβάλλοντα είναι συχνά ευαίσθητα σε ασήμαντες λύσεις με σενάριο , απαιτούμε τη χρήση ενός στοχαστικού

"sticky frame skip". Το Sticky frame skip προσθέτει μια μικρή τυχαιότητα στις ενέργειες του πράκτορα. δεν αλλάζει άμεσα παρατηρήσεις ή ανταμοιβές. Όπως και το τυπικό πλαίσιο παράλειψης, το κολλώδες πλαίσιο παράλειψης εφαρμόζει n ενέργειες σε $4n$ καρέ. Ωστόσο, για κάθε ενέργεια, την καθυστερούμε κατά ένα πλαίσιο με πιθανότητα 0.25, εφαρμόζοντας την προηγούμενη ενέργεια για αυτό το πλαίσιο. Το παρακάτω διάγραμμα δείχνει ένα παράδειγμα μιας ακολουθίας δράσης με παράλειψη κολλώδους πλαισίου:



Σχήμα 5: Sticky Frame

4.1.6 Λήξεις επεισοδίων

Η εμπειρία στο παιχνίδι χωρίζεται σε επεισόδια, τα οποία αντιστοιχούν περίπου στη ζωή. Στο τέλος κάθε επεισοδίου, το περιβάλλον επαναφέρεται στην αρχική του κατάσταση αποθήκευσης. Τα επεισόδια μπορούν να λήξουν σε τρεις 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 παίκτης χάνει μια ζωή
- 4500 χρονικά βήματα έχουν περάσει στο τρέχον επεισόδιο. Αυτό ισοδυναμεί με περίπου 5 λεπτά χρόνου παιχνιδιού.

Λάβετε υπόψη ότι το benchmark μας παραλείπει τις μάχες αφεντικού που συχνά πραγματοποιούνται στο τέλος ενός επιπέδου. Για επίπεδα με μάχες αφεντικού, η τελική μας κατάσταση ορίζεται ως οριζόντια μετατόπιση που πρέπει να φτάσει ο πράκτορας πριν από τον αγώνα του αφεντικού. Αν και οι μάχες αφεντικού θα μπορούσαν να είναι ένα ενδιαφέρον πρόβλημα για επίλυση, είναι αρκετά διαφορετικές από το υπόλοιπο του παιχνιδιού. Δεν συμπεριλαμβάνονται έτσι ώστε να μπορούμε να επικεντρωθούμε περισσότερο στην εξερεύνηση, την πλοήγηση και την ταχύτητα.

4.1.7 Παρατηρήσεις και Δράσεις

Ένα gym-retro περιβάλλον παράγει μια παρατήρηση στην αρχή κάθε χρονικού βήματος. Αυτή η παρατήρηση είναι πάντα μια εικόνα RGB 24 bit, αλλά οι διαστάσεις ποικίλλουν ανάλογα με το παιχνίδι. Για τον Sonic, οι εικόνες της οθόνης έχουν πλάτος 320 εικονοστοιχεία και ύψος 224 εικονοστοιχεία.

Σε κάθε χρονικό βήμα, ένας πράκτορας παράγει μια ενέργεια που αντιπροσωπεύει έναν συνδυασμό κουμπιών στην κονσόλα παιχνιδιών. Οι ενέργειες κωδικοποιούνται ως δυαδικά διανύσματα, όπου 1 σημαίνει ¨πιεσμένο' και 0 σημαίνει ¨μη πατημένο¨. Για τα παιχνίδια Sega Genesis, ο χώρος δράσης περιέχει τα ακόλουθα κουμπιά: B, A, MODE, START, UP, DOWN, LEFT, RIGHT, C, Y, X, Z.

Ένα μικρό υποσύνολο όλων των πιθανών συνδυασμών κουμπιών έχει νόημα στο Sonic. Στην πραγματικότητα, υπάρχουν οκτώ χρήσιμοι συνδυασμοί κουμπιών: {}, {LEFT}, {RIGHT}, {LEFT, DOWN}, {RIGHT, DOWN}, {DOWN}, {DOWN, B}, {B}

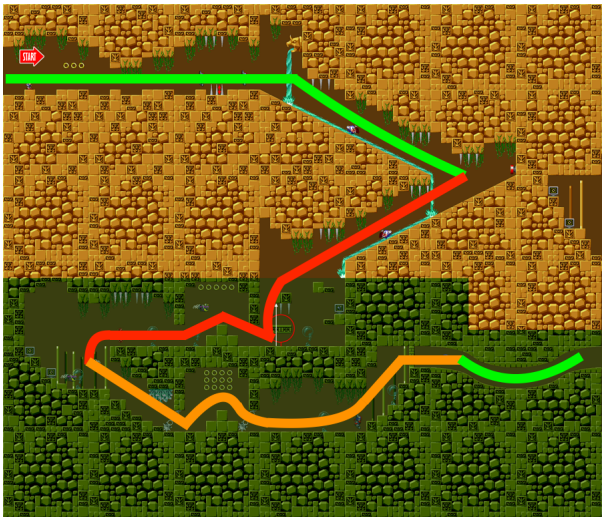
4.1.8 Συνάρτηση Ανταμοιβής

Κατά τη διάρκεια ενός επεισοδίου, οι πράκτορες ανταμείβονται έτσι ώστε η αθροιστική ανταμοιβή ανά πάσα στιγμή να είναι ανάλογη με την οριζόντια μετατόπιση από την αρχική θέση του παίκτη. Έτσι, το να πηγαίνει δεξιά πάντα αποδίδει θετική ανταμοιβή, ενώ το αριστερό πάντοτε αποδίδει αρνητική ανταμοιβή. Αυτή η συνάρτηση ανταμοιβής είναι σύμφωνη με την ολοκληρωμένη μας κατάσταση, η οποία βασίζεται στην οριζόντια μετατόπιση στο επίπεδο.

Η ανταμοιβή αποτελείται από δύο στοιχεία: μια οριζόντια αντιστάθμιση και ένα μπόνους ολοκλήρωσης. Η οριζόντια επιβράβευση αντιστάθμισης ομαλοποιείται ανά επίπεδο, έτσι ώστε η συνολική επιβράβευση ενός πράκτορα να είναι 9000 εάν φτάσει στην προκαθορισμένη οριζόντια μετατόπιση που σηματοδοτεί το τέλος του επιπέδου. Με αυτόν τον τρόπο, είναι εύκολο να συγκρίνουμε βαθμολογίες σε επίπεδα διαφορετικού μήκους. Το μπόνους ολοκλήρωσης είναι 1000 για την άμεση επίτευξη του επιπέδου και πέφτει γραμμικά στο μηδέν στα 4500 βήματα. Με αυτόν τον τρόπο, οι πράκτορες ενθαρρύνονται να ολοκληρώσουν τα επίπεδα όσο το δυνατόν γρηγορότερα.

Δεδομένου ότι η συνάρτηση επιβράβευσης είναι πυκνή, οι αλγόριθμοι RL όπως το PPO και το DQN μπορούν εύκολα να σημειώσουν πρόοδο σε νέα επίπεδα. Ωστόσο, οι άμεσες ανταμοιβές μπορεί να είναι παραπλανητικές. Είναι συχνά απαραίτητο να πάμε προς τα πίσω για παρατεταμένα χρονικά διαστήματα (6).

Χρησιμοποιούμε την προεπεξεργασία ανταμοιβής, έτσι ώστε οι πράκτορές μας να μην τιμωρούνται για το ότι πηγαίνουν πίσω. Σημειώστε, ωστόσο, ότι η προεπεξεργασμένη ανταμοιβή δεν δίνει καμία πληροφορία σχετικά με το πότε ή πώς πρέπει να πάει ο πράκτορας προς τα πίσω.



Σχήμα 6: Μία τροχειά σε μία πίστα. Στο αρχικό πράσινο τμήμα, ο πράκτορας κινείται προς τα δεξιά, παίρνοντας θετική ανταμοιβή.Στο κόκκινο τμήμα, ο πράκτορας πρέπει να κινηθεί προς τα αριστερά, παίρνοντας αρνητική ανταμοιβή. Κατά τη διάρκεια του πορτοκαλί τμήματος, ο πράκτορας κινείται και πάλι δεξιά, αλλά η αθροιστική ανταμοιβή του δεν είναι ακόμα τόσο υψηλή όσο ήταν μετά το αρχικό πράσινο τμήμα. Στο τελικό πράσινο τμήμα, ο πράκτορας βελτιώνει τελικά τη σωρευτική ανταμοιβή του μετά το αρχικό πράσινο τμήμα.

4.1.9 Εκτίμηση (και ο OpenAI Retro διαγωνισμός)

Το 2018, η OpenAi ξεκίνησε έναν διαγωνισμό -ο οποίος χρησίμευσε ως έμπνευση πίσω από αυτήν την εργασία- αφού κυκλοφόρησε το περιβάλλον Sonic, με την πρόθεση να παρέχει ένα περιβάλλον που είναι βέλτιστο για να δοκιμάσει κανείς την δυνατότητα μεταφοράς εκμάθησης με πολλούς αλγόριθμους RL. Σε αυτήν την ενότητα περιγράφουμε τους κανόνες που ακολουθήθηκαν καθ ΄όλη τη διάρκεια του διαγωνισμού και αυτής του εργασίας.

Γενικά, όλα τα benchmarks πρέπει να παρέχουν κάποιο είδος μέτρησης απόδοσης. Για το Sonic, αυτή η μέτρηση έχει τη μορφή «μέσης βαθμολογίας» όπως μετράται σε όλα τα επίπεδα στο σύνολο δοκιμών. Ακολουθούν τα γενικά βήματα για την αξιολόγηση ενός αλγορίθμου στο Sonic:

1. Κατά την εκπαίδεση, χρησιμοποιούμε το σετ εκπαίδευσης όσο θέλουμε.

2. Κατά τη διάρκεια της δοκιμής, παίζουμε κάθε επίπεδο δοκιμής για 1 εκατομμύριο βήματα. Παίζουμε κάθε επίπεδο δοκιμής ξεχωριστά. Δεν επιτρέπουμε τη ροή πληροφοριών μεταξύ των επιπέδων δοκιμής. Μπορούμε να χρησιμοποιήσουμε πολλαπλά αντίγραφα κάθε περιβάλλοντος (όπως γίνεται σε παράλληλους αλγόριθμους όπως το A3C [2])

3. Για κάθε αξιολόγηση ενός εκατομμυρίου χρονοβημάτων, υπολογίζουμε τον μέσο όρο της συνολικής ανταμοιβής ανά επεισόδιο σε όλα τα επεισόδια. Αυτό δίνει μια μέση βαθμολογία ανά επίπεδο.

4. Υπολογίζουμε τον μέσο όρο των μέσων βαθμολογιών για όλα τα επίπεδα δοκιμής, δίνοντας μια συνολική μέτρηση απόδοσης.

Η πιο σημαντική πτυχή αυτής της διαδικασίας είναι το χρονικό όριο για κάθε επίπεδο δοκιμής. Στο καθεστώς άπειρου χρονικού διαστήματος, δεν υπάρχει κανένας ισχυρός λόγος να πιστεύουμε ότι η μετα-μάθηση ή η μεταφορά μάθησης είναι απαραίτητη. Ωστόσο, στο καθεστώς περιορισμένου χρονικού διαστήματος, η μεταφορά εκμάθησης ενδέχεται να είναι απαραίτητη για γρήγορη επίτευξη καλής απόδοσης.

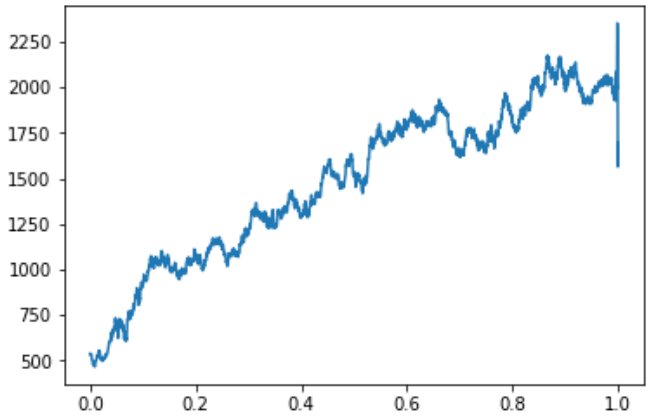
4.2 Πειραματικά Αποτελέσματα

4.2.1 Εισαγωγή

Σε αυτήν την ενότητα παρουσιάζουμε τα αποτελέσματα των αλγορίθμων που συζητήθηκαν προηγουμένως. Πιο συγκεκριμένα, παρουσιάζουμε τόσο το PPO όσο και το Rainbow ως εκπαιδευμένο αποκλειστικά σε κάθε επίπεδο δοκιμής ανεξάρτητα και στη συνέχεια δείχνουμε τα αποτελέσματά τους στα εν λόγω επίπεδα δοκιμής μετά την εκπαίδευσή τους, από κοινού, στα επίπεδα του συνόλου εκπάιδευσης. Τέλος, συγκρίνουμε αυτά τα αποτελέσματα με αυτά του OpenAI και εξάγουμε συμπεράσματα σχετικά με την αποτελεσματικότητα των αλγοριθμων στο περιβάλλον ανεξάρτητα από την υλοποίηση.
Θα προχωρήσουμε, έχοντας κατά νου ότι η μέση βαθμολογία ανθρώπων ήταν 7438.2, όπου σύμφωνα με την περιγραφή του διαγωνισμού: ΄.. είχαμε τέσσερα άτομα να παίζουν κάθε επίπεδο δοκιμής για μία ώρα. Πριν δουν τα επίπεδα δοκιμής, το καθένα άτομα είχε δύο ώρες για να εξασκηθεί στα επίπεδα εκπαίδευσης.΄

4.2.2 Rainbow

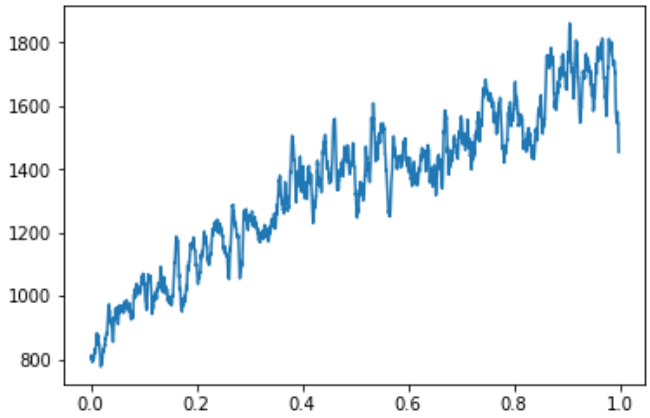
Σε αυτήν την ενότητα θα δημοσιεύσουμε τα αποτελέσματα του προαναφερθέντος αλγορίθμου Rainbow. Το χρησιμοποιήσαμε ανεξάρτητα σε κάθε επίπεδο δοκιμής και μετρήσαμε τον μέσο όρο της απόδοσης σε κάθε ένα. Οι περισσότερες από τις παραμέτρους είναι ίδιες όπως στην αρχική έκδοση [4] με κάποιες διαφορές. Κατάρχάς, ορίσαμε το $V_{max} = 200$ για να συνυπολογίσουμε την κλίμακα ανταμοιβής του Sonic και χρησιμοποιούμε μέγεθος replay buffer 0,1M αντί 1M λόγω πόρων. Επίσης, μειώσαμε το χρόνο ανάμεσα σε διαδοχικές ενημερώσεις του target DQN σε 4500 από 32000. Προεπεξεργαστήκαμε επίσης τις παρατηρήσεις και τις μετατρέψαμε σε γκρίζα κλίμακα για να υποτιμήσουμε την είσοδο σε μια στοίβα τεσσάρων $84x84$ εικόνων ενός καναλιού. Ο μέσος όρος καθ΄ όλη την εκπαίδευση ήταν 1833.4 και το τελικό σκορ 2221.



Σχήμα 7: Καμπύλης εκμάθησης ,για Rainbow στα test levels.
Μέσος όρος : 1833.4, τελικό : 2221. Μέσο κέρδος στα test levels - βήματα εκπαίδευσης σε εκατομύρρια

4.2.3 PPO

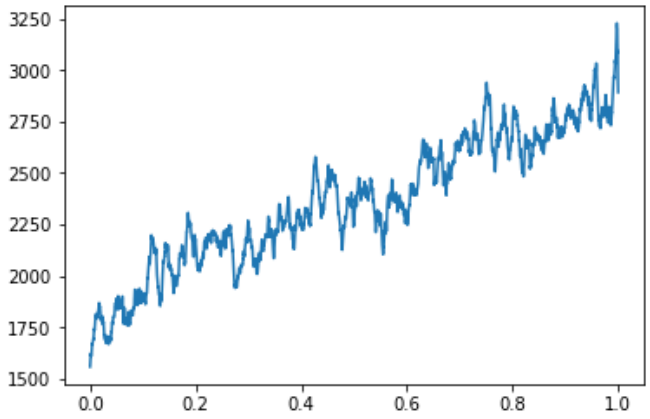
Για PPO χρησιμοποιούμε τους ίδιους χώρους δράσης και παρατήρησης όπως και για το Rainbow, καθώς και την ίδια προεπεξεργασία ανταμοιβής. Για τα πειράματά μας, κλιμακώσαμε τα οφέλη με έναν μικρό σταθερό παράγοντα προκειμένου να φέρουμε τα πλεονεκτήματα σε ένα κατάλληλο εύρος για νευρωνικά δίκτυα. Η αρχιτεκτονική του CNN είναι ίδια με αυτήν που χρησιμοποιήθηκε στην αρχική έκδοση PPO για το Atari, [5]. Μερικές διαφορές: ο ορίζοντας ,που σηματοδοτεί το τέλος του επεισοδίου και τον υπολογισμό των gradients, ήταν 10000. Η παράμετρος αποκοπής (clipping parameter ϵ) ακολούθησε μια γραμμικά ανόπτηση, που σημαίνει ότι ξεκίνησε στα 0,2 και έπεσε γραμμικά στο 0 κατά την εκπαίδευση. Επιπλέον, σε αντίθεση με την αρχική εφαρμογή χρησιμοποιήσαμε μια υλοποίηση ενός νήματος. Ο μέσος όρος καθ΄ όλη την εκπαίδευση ήταν 1347.5 και το τελικό σκορ 1688.4.



Σχήμα 8: Καμπύλης εκμάθησης για PPO στα test levels.
Μέσος όρος : 1347.5 , τελικό : 1688.4. Μέσο κέρδος στα test levels - βήματα εκπαίδευσης σε εκατομύρρια

4.2.4 joint PPO

Ενώ η προηγούμενη ενότητα αξιολογεί το PPO χωρίς μετα-μάθηση, αυτή η ενότητα διερευνά την ικανότητα του PPO να μεταφέρει από τα επίπεδα εκπαίδευσης στα επίπεδα δοκιμής. Χρησιμοποιούμε έναν απλό αλγόριθμο κοινής εκπαίδευσης, που σημαίνει ότι εκπαιδεύσαμε ένα μοντέλο σε όλα τα επίπεδα εκπαίδευσης, αλλάζοντας τυχαία μεταξύ τους στο τέλος κάθε επεισοδίου και το χρησιμοποιούσαμε ως αρχικοποίηση στα επίπεδα δοκιμής. Οι παράμετροι ήταν οι ίδιες με πριν με κάποιες διαφορές. Η διάρκεια της εκπαίδευσης ήταν 5M στο σύνολο της προπόνησης στο σύνολό της και ο ορίζοντας ορίστηκε σε 128 για την επακόλουθη προπόνηση στα επίπεδα δοκιμής για να αυξηθεί ο αριθμός των ενημερώσεων εντός του περιορισμού του 1M. Ο μέσος όρος στα test levels ήταν 2354.4 με τελικό 3062.1.

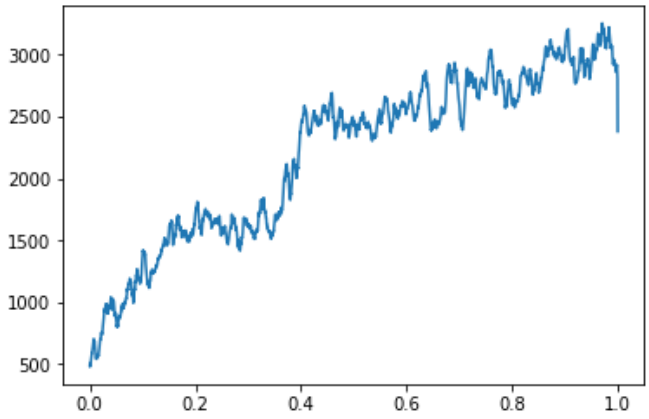


Σχήμα 9: Καμπύλης εκμάθησης για PPO στα test levels, προεκπαιδευμένο στα train levels. Μέσος όρος : 2354.4, τελικό : 3062.1. Μέσο κέρδος στα test levels - βήματα εκπαίδευσης σε εκατομύρρια

Τα αποτελέσματα είναι σημαντικά λαμβάνοντας υπόψη δύο σημεία. Πρώτον, όπως φαίνεται από το παραπάνω σχήμα, ο αλγόριθμος είχε ήδη υψηλή βαθμολογία στα επίπεδα δοκιμής μόλις ξεκίνησε η διαδικασία, σε σύγκριση με την περίπτωση μη μετα-μάθησης (4.2.3) όπου ξεκίνησε φυσικά πολύ χαμηλά. Και δεύτερον, η αύξηση ήταν 75% κατά μέσο όρο με μόλις 5 εκατομμύρια βήματα προ-εκπαίδευσης.Θα δούμε αργότερα ότι η εκπαιδεύοντας ακόμα περισσότερο στο training set μπορεί να έχει ακόμη καλύτερα αποτελέσματα.

4.2.5 joint Rainbow

Δεδομένου ότι το Rainbow ξεπερνά το PPO χωρίς κοινή εκπαίδευση, είναι φυσικό να αναρωτηθούμε αν το Joint Rainbow ξεπερνά με ανάλογο τρόπο το joint PPO. Παραδόξως, τα πειράματά μας δείχνουν ότι αυτό δεν ισχύει. Ο αλγόριθμος ξεκίνησε τόσο χαμηλά όσο η μεμονωμένη περίπτωση και παρόλο που είχε μια αξιοσημείωτη τελική διαφορά, δεν ήταν τόσο αισθητή, κατά μέσο όρο (16.3%), όπως η περίπτωση joint PPO. Η διάταξη ήταν η ίδια όπως πριν και τα αποτελέσματα ήταν : μέσος όρος 2192.2 (+300 αύξηση, περίπου 16.3%), τελικό 2942.



Σχήμα 10: Καμπύλης εκμάθησης για Rainbow στα test levels, προεκπαιδευμένο στα train levels. Μέσος όρος : 2192.2, τελικό : 2942. Μέσο κέρδος στα test levels - βήματα εκπαίδευσης σε εκατομύρρια

4.2.6 Σύγκριση με τα αποτελέσματα OpenAI

Όπως αναφέρθηκε προηγουμένως, η ιδέα που ενέπνευσε αυτή την εργασία είναι από το διαγωνισμό του OpenAI [3]. Τα αποτελέσματα αναλυτικά αναφέρονται στο κεφάλαιο των πινάκων. Συνοπτικά οι μέσοι όροι: Σε αντίθεση με τις

Algorithm	Score	OpenAI Score
PPO	1347.5	1488.8
joint PPO	2354.4	3127.9
Rainbow	1833.4	2748.6
joint Rainbow	2192.2	2969.2

διαφορετικές υλοποιήσεις μπορούμε να εξαγάγουμε ορισμένα ενημερωτικά συμπεράσματα σχετικά με τις παραμέτρους και τον τρόπο με τον οποίο οι διαφορετικές υλοποιήσεις επηρεάζουν το τελικό αποτέλεσμα, ενώ τα κύρια χαρακτηριστικά της συνολικής διαδικασίας παραμένουν ανέπαφα.

- Πρώτα απ ΄όλα, το απλό PPO δεν έχει ιδιαίτερη διαφορά μεταξύ των δύο. Η μικρή απόκλιση (1347 έναντι 1450) μπορεί να αποδοθεί στον διαφορετικό ορίζοντα (10000 έναντι 8192). Γενικά ένας μικρότερος ορίζοντας σημαίνει περισσότερες ενημερώσεις.

- Η δεύτερη σύγκριση θα ήταν μεταξύ των δύο joint PPO. Υπάρχουν τρεις εξαιρετικά μεγάλες διαφορές που αντιπροσωπεύουν τη διαφορά στο σκορ. Το πρώτο είναι το ύψος της εκπαίδευσης στο σετ εκπαίδευσης. Το δικό μας ήταν για 5M, ενώ το OpenAI ήταν για οπουδήποτε μεταξύ 50M και 400M με τη σημείωσή τους ότι πάνω από 50M δεν παρατηρήθηκε βελτίωση. Το δεύτερο είναι το γεγονός ότι ο πράκτορας τους εκπαιδεύτηκε παράλληλα από 188 workers. Είναι αποδεδειγμένο στο [2] ότι η οικογένεια των αλγορίθμων PG επωφελείται σημαντικά από τον παραλληλισμό. Και τέλος, στη συνέχεια έκαναν fine tuning των παραμέτρων σε κάθε test level.

- Η διαφορά μεταξύ των αλγορίθμων Rainbow ήταν πιο εντυπωσιακή. Η διαφορά στις παραμέτρους ήταν στο PER που ήταν 0,1M για εμάς και 0,5M για το OpenAI. Μπορούμε να καταλήξουμε στο συμπέρασμα ότι, παρά την προτεραιότητα, η μεγαλύτερη μνήμη επιτρέπει μεγαλύτερη ποικιλία στις παρτίδες (batch). Το δεύτερο ήταν η διαφορά στο μέγεθος παρτίδας (32 έναντι 256).

- Τέλος, παρόμοιες διαφορές υπήρχαν για το joint Rainbow, που σημαίνει υψηλότερο μέγεθος παρτίδας, μνήμης, χρόνο εκπαίδευσης (50M και περισσότερο), fine tuning ανά test level, παράλληλους workers. Παρόλο που η παραλληλοποίηση δεν βελτιώνει την οικογένεια DQN όσο και την PG, ωστόσο αυξάνει την ποικιλία των δειγμάτων στη μνήμη.

Παρά τα παραπάνω, τα κύρια συμπεράσματα και ο σκοπός αυτής της σύγκρισης είναι ότι οι κύριες παρατηρήσεις σχετικά με την αποτελεσματικότητα αυτών των

αλγορίθμων στο πλαίσιο του περιβάλλοντος Sonic καθώς και την ικανότητά τους να μεταφέρουν τη μάθηση, είναι οι ίδιες και τα δύο πειράματα, αν και σε διαφορετικό μέγεθος. Το Rainbow ήταν ανώτερο από το PPO στην εκπαίδευση αποκλειστικά στα επίπεδα των δοκιμών, αλλά το PPO επωφελήθηκε από την εκμάθηση της μεταφοράς.

5 Συμπεράσματα και τελευταίες παρατηρήσεις

5.1 Σύνοψη και αξιολόγηση

Συμπερασματικά, έχουμε δοκιμάσει την αποτελεσματικότητα δύο αρχετά σύγχρονων αλγορίθμων, δηλαδή PPO και Rainbow, από δύο διαφορετικές οικογένειες (DQN και Policy Gradients). Αξιολογήσαμε την αποτελεσματικότητά τους στο σημείο αναφοράς Sonic the Hedgehog. Στη συνέχεια αξιολογήσαμε την αποτελεσματικότητά τους στη μεταφορά μάθησης. Παρατηρήσαμε ότι στην απλή περίπτωση ενός worker χωρίς προεκπαίδευση το Rainbow ξεπερνά το PPO αλλά με παραλληλισμό και ειδικά με προεπεξεργασία τα αποτελέσματα αντιστρέφονται. Τα αποτελέσματά μας αφήνουν πολλά περιθώρια βελτίωσης, ειδικά επειδή τα καλύτερα αποτελέσματα μάθησης μεταφοράς δεν είναι πολύ καλύτερα από τα καλύτερα αποτελέσματα που μαθαίνουμε από το μηδέν.

5.2 Παρατηρήσεις, Συμπεράσματα και μελλοντικές κατευθήνσεις

Παρατηρώντας τη συμπεριφορά του πράκτορα στο παιχνίδι, παρατηρήσαμε ότι παρεμποδίζεται σοβαρά από τη συνάρτηση σκορ. Θυμηθείτε ότι η συνάρτηση βαθμολογίας επιβραβεύει τον πράκτορα για το πόσο μακριά μπορεί να προχωρήσει οριζόντια. Αυτό ήταν τελικά ένα πρόβλημα όπου για να φτάσει στο τέλος της πίστας, ο πράκτορας έπρεπε να κάνει μια μεγάλη παράκαμψη, είτε αριστερά είτε προς τα πάνω ή προς τα κάτω.

Για παράδειγμα, υπάρχουν πολλά στάδια όπου ο πράκτορας αγνόησε έναν ορατό ανελκυστήρα που ήταν ελαφρώς πίσω του μέσα στο καρέ και αντ άυτού κόλλησε, πιέζοντας τον τοίχο προς τα δεξιά. Σε άλλες περιπτώσεις, δεν μπόρεσε να αναβάλει τη μετάβαση προς τα εμπρός και να οπισθοδρομήσει σε μια προσπάθεια να αυξήσει την απαραίτητη ταχύτητα για να ξεπεράσει τα εμπόδια.

Μια άλλη σημείωση είναι ότι οι παρατηρήσεις δεν έχουν μνήμη, εστιάζοντας μόνο στα 4 τελευταία καρέ, δυνητικά οδηγώντας τον πράκτορα να συγχέεται μεταξύ των levels που μοιράζονται πολλά οπτικά χαρακτηριστικά, π.χ. acts εντός της ίδιας zone.

Αυτά θέτουν δύο ενδιαφέροντα ερωτήματα. Εάν πρέπει να εφαρμόσουμε ένα βαθύτερο νευρωνικό δίκτυο που εφαρμόζει RNNs, Transformers και άλλα για να δημιουργήσουμε καλύτερη απομνημόνευση, αλγόριθμους που δίνουν προτεραιότητα στην εξερεύνηση παρά τη συνάρτηση βαθμολογίας και που μπορούν να διαμορφώσουν ένα μακροπρόθεσμο σχέδιο για να ξεπεράσουν ένα εμπόδιο.

Παρόλο που πιστεύουμε ότι το Sonic benchmark είναι ένα βήμα προς τη σωστή κατεύθυνση, ενδέχεται να μην επαρκεί για την εξερεύνηση της μετα-μάθησης, της μεταφοράς μάθησης και της γενίκευσης στο RL. Ακολουθούν ορισμένα πιθανά προβλήματα με αυτό το benchmark :

- Μπορεί να είναι δυνατή η επίλυση ενός Sonic level σε πολύ λιγότερα από 1M βήματα χωρίς μεταφορά μάθησης.

- Hacks ειδικά για το Sonic μπορεί να ξεπεράσουν τις γενικές προσεγγίσεις μετα-μάθησης.
- Οι στρατηγικές εξερεύνησης που λειτουργούν καλά στο Sonic ενδέχεται να μην γενικεύονται πέρα από αυτό.
- Ο έλεγχος ενός επιπέδου Sonic περιλαμβάνει κάποιο βαθμό απομνημόνευσης. Αλγόριθμοι που είναι καλοί σε λίγη απομνημόνευση μπορεί να μην είναι καλοί σε άλλες εργασίες.

Αναφορές

Reinforcement Learning: An Introduction, by Richard S. Sutton and Andrew G. Barto

Asynchronous Methods for Deep Reinforcement Learning, Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, Koray Kavukcuoglu

"Gotta Learn Fast: A New Benchmark for Generalization in RL", Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, John Schulman, OpenAI

Rainbow: Combining Improvements in Deep Reinforcement Learning, Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, David Silver

Proximal Policy Optimization Algorithms, John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov, OpenAI

Playing Atari with Deep Reinforcement Learning, Volodymyr Mnih, Koray Kavukcuoglu, David Silver,Daan Wierstra, Alex Graves, Ioannis Antonoglou , Martin Riedmiller, DeepMind Technologies

Deep Reinforcement Learning with Double Q-learning, Hado van Hasselt and Arthur Guez and David Silver

Prioritized Experience Replay, Tom Schaul, John Quan, Ioannis Antonoglou and David Silver

Noisy Networks for Exploration, Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Remi Munos, Demis Hassabis, Ian Osband, Alex Graves,Olivier Pietquin,Vlad Mnih,Charles Blundell, Shane Legg

Dueling Network Architectures for Deep Reinforcement Learning, Ziyu Wang,Tom Schaul,Matteo Hessel,Hado van Hasselt,Marc Lanctot, Nando de Freitas

A Distributional Perspective on Reinforcement Learning Marc G. Bellemare, Will Dabney, Rémi Munos *The Arcade Learning Environment: An Evaluation Platform for General Agents* Marc G. Bellemare, Yavar Naddaf, Joel Veness, Michael Bowling

6 Πίνακες Αποτελεσμάτων

State	Score	Final Score
SpringYardZone Act1	510.4	743.5
GreenHillZone Act2	1946.6	2380.1
StarLightZone Act3	2121.8	2076.5
ScrapBrainZone Act1	810.6	1122.2
MetropolisZone Act3	900.4	1373.8
HillTopZone Act2	1272.6	1938.3
CasinoNightZone Act2	2578.2	3264.4
LavaReefZone Act1	905.2	1635.4
FlyingBatteryZone Act2	1579	1694.4
HydrocityZone Act1	642.6	790.4
AngelIslandZone Act2	1554.9	1553.8
Aggregate	1347.5	1688.4

1: PPO results

State	Score	Final Score
SpringYardZone Act1	564.2	644.2
GreenHillZone Act2	2477.6	2655.7
StarLightZone Act3	2134.4	2519
ScrapBrainZone Act1	1162	2190.8
MetropolisZone Act3	1007.6	1058.6
HillTopZone Act2	2408	3173.1
CasinoNightZone Act2	2517.8	2343.6
LavaReefZone Act1	885.8	683.9
FlyingBatteryZone Act2	1105.8	1305.7
HydrocityZone Act1	622.8	433.5
AngelIslandZone Act2	1491.3	2298.3
Aggregate	1488.8	1755.1

2: PPO results-OpenAI

State	Score	Final Score
SpringYardZone Act1	1486.5	2158.1
GreenHillZone Act2	2979.6	3502.4
StarLightZone Act3	2840.6	2970.2
ScrapBrainZone Act1	1131.5	1250.7
MetropolisZone Act3	1208.8	1918.5
HillTopZone Act2	3141.7	3651.1
CasinoNightZone Act2	6799.6	8697.3
LavaReefZone Act1	1862.4	3673.6
FlyingBatteryZone Act2	1573.1	1718.3
HydrocityZone Act1	1088.2	2039.2
AngelIslandZone Act2	1786.1	2103.7
Aggregate	2354.4	3062.1

3: joint PPO results

State	Score	Final Score
SpringYardZone Act1	2992.9	4663.4
GreenHillZone Act2	8769.3	8921.2
StarLightZone Act3	1445.3	2636.7
ScrapBrainZone Act1	1634.6	2112
MetropolisZone Act3	1409.5	2004.3
HillTopZone Act2	4289.9	4688.6
CasinoNightZone Act2	5410.2	6142.4
LavaReefZone Act1	2409	3076
FlyingBatteryZone Act2	1513.3	1748
HydrocityZone Act1	1249.8	2821.7
AngelIslandZone Act2	3283	4375.3
Aggregate	3127.9	3926.3

4: joint PPO results-OpenAI

State	Score	Final Score
SpringYardZone Act1	1054.4	1923.2
GreenHillZone Act2	2890.1	2891.8
StarLightZone Act3	2660	2960.3
ScrapBrainZone Act1	335	282.7
MetropolisZone Act3	1059.7	1807.7
HillTopZone Act2	1500	1661.6
CasinoNightZone Act2	4191.5	4186.1
LavaReefZone Act1	1470.9	2391.3
FlyingBatteryZone Act2	1667.9	1856.4
HydrocityZone Act1	2520	2653
AngelIslandZone Act2	817.9	1817.3
Aggregate	1833.4	2221

5: Rainbow results

State	Score	Final Score
SpringYardZone Act1	1787.6	3861
GreenHillZone Act2	6332	6817.2
StarLightZone Act3	2421.9	2680.3
ScrapBrainZone Act1	879.1	2050
MetropolisZone Act3	1178.1	2278.8
HillTopZone Act2	2847.8	3432.7
CasinoNightZone Act2	6045.2	8607.9
LavaReefZone Act1	2623.6	2908.5
FlyingBatteryZone Act2	1657.5	2195.4
HydrocityZone Act1	886.4	867.2
AngelIslandZone Act2	3576	5070.1
Aggregate	2748.6	3706.3

6: Rainbow results-OpenAI

State	Score	Final Score
SpringYardZone Act1	523.7	562.8
GreenHillZone Act2	7478.3	9168.8
StarLightZone Act3	2640.8	3102.9
ScrapBrainZone Act1	1539.5	2637.4
MetropolisZone Act3	1014.6	2217.1
HillTopZone Act2	2231.4	1964
CasinoNightZone Act2	4038.9	5679.3
LavaReefZone Act1	1347.1	2679.5
FlyingBatteryZone Act2	1526.5	1688
HydrocityZone Act1	738.9	867.6
AngelIslandZone Act2	1035	1794.1
Aggregate	2192.2	2942

7: joint Rainbow

State	Score	Final Score
SpringYardZone Act1	2661	4090.1
GreenHillZone Act2	6106.8	6793.5
StarLightZone Act3	1813.7	2533.8
ScrapBrainZone Act1	983.5	2075
MetropolisZone Act3	1340.6	1843.2
HillTopZone Act2	2378.4	3531.3
CasinoNightZone Act2	7877.7	8851.2
LavaReefZone Act1	2753.6	2959.7
FlyingBatteryZone Act2	2110.2	2585.7
HydrocityZone Act1	865	867.2
AngelIslandZone Act2	3770.5	4615.1
Aggregate	2969.2	3704.2

8: joint Rainbow-OpenAI

State	PPO	OpenAI PPO	No Noisy	Rainbow	OpenAI Rainbow
SpringYardZone Act3	1877.3	2608.1	2259.4	1872.7	2029.6
SpringYardZone Act2	1944.9	9306.8	1230.5	1062.6	3162.3
GreenHillZone Act3	917.7	9878.5	1006.8	400.6	5481.3
GreenHillZone Act1	4227.8	7116	2525	1140.9	4164.7
StarLightZone Act2	2168.5	8336.1	3454.4	1540.4	7105.5
StarLightZone Act1	3457	6363.6	1985.4	2367.5	4558.9
MarbleZone Act2	1845.9	1620.6	1629	1576	1615.7
MarbleZone Act1	3130.6	5007.8	1779.9	1493.5	4127
MarbleZone Act3	2590.5	2054.4	2256.4	2125.9	1595.1
ScrapBrainZone Act2	1244.6	1403.7	963.6	925	692.6
LabyrinthZone Act2	2488.6	1337.9	2594.3	2479.5	1420.8
LabyrinthZone Act1	1838.4	5041.4	1721.1	805.7	3005.3
LabyrinthZone Act3	1681.2	1918.7	1204	970.9	1458.7
EmeraldHillZone Act1	6832.6	9870.7	8021	6680.7	9273.4
EmeraldHillZone Act2	5482.2	9901.6	4590.9	9010.6	9410.1
ChemicalPlantZone Act2	2848.7	2586.8	2833.6	1420.1	2840.4
ChemicalPlantZone Act1	3786.9	9825	3305.1	3376.8	4483.5
MetropolisZone Act1	1522.1	1102.8	636.1	1099.2	388.9
MetropolisZone Act2	2230.5	6666.7	1334.2	1707	3048.6
OilOceanZone Act1	1441.5	4938.8	995	615	1998.8
OilOceanZone Act2	1701.9	6964.9	1833.1	930.9	3613.7
MysticCaveZone Act2	706.8	6189.6	1615.2	596.9	4359.4
MysticCaveZone Act1	1157.5	6755.9	1201	856.2	1606.8
HillTopZone Act1	736.5	4074.2	758.5	82.5	778
CasinoNightZone Act1	2141.4	9378.8	1729.3	1111.6	2165.7
WingFortressZone	920.7	3109.2	2251.3	1064.7	3004.6
AquaticRuinZone Act2	3007.9	8676	2949.1	1973.3	4752.7
AquaticRuinZone Act1	3807.9	9879.8	2858	2847.7	5382.3
LavaReefZone Act2	475.2	2155.1	545.4	313	820.3
CarnivalNightZone Act2	2298.3	2688.2	2354.4	2224.7	2613.7
CarnivalNightZone Act1	4239.6	4429.5	5139.1	1582.9	3554.8
MarbleGardenZone Act1	1858.4	3760	2479.4	1911.3	2733.2
MarbleGardenZone Act2	317.5	1366.4	602.2	317.5	180.7
MushroomHillZone Act2	1645.1	6549.6	1440.3	1190.7	2869.1
MushroomHillZone Act1	2789.6	3210.2	2668.5	2607.2	2076
DeathEggZone Act1	935.7	3332.5	1435.6	645.9	2334.3
DeathEggZone Act2	598.9	3141.5	1178.8	752.4	3197.8
FlyingBatteryZone Act1	1302.2	1642.4	1302.2	1283	711.8
SandopolisZone Act1	553.4	2548.1	278	274.2	1475.3
SandopolisZone Act2	586.6	1087.5	528	529.7	539.9
HiddenPalaceZone	7656.5	9918.3	7794.6	4485.5	9308.9
HydrocityZone Act2	20.9	4756.8	788	20	825.7
IcecapZone Act1	4855.5	5389.9	5283.1	4850.9	5507
IcecapZone Act2	2268.9	6819.4	2569.7	73.5	3198.2
AngelIslandZone Act1	3075.5	9668.2	1914.7	2048.2	4765.6
LaunchBaseZone Act2	2381.2	1836	2416.7	2060	1850.1
LaunchBaseZone Act1	1967.3	2714	1224.4	1356.9	2044.5
Aggregate	2288.6	5083.6	2201.4	1716.2	3151.7

9: All Training Results