ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Σχολή Τεχνολογίας

Τμήμα Ψηφιακών Συστημάτων

Βαθιά Ενισχυτική Μάθηση για Ανταγωνιστικά Παίγνια

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Δημήτριος Καλαφάτης (ΑΜ: 00351)

Επιβλέπων: ***Φώτιος Κόκκορας, Επίκουρος Καθηγητής***

ΛΑΡΙΣΑ, Ιούνιος 2024

UNIVERSITY OF THESSALY

School of Technology

Digital Systems Department

Deep Reinforcement Learning for Adversarial Games

BACHELOR THESIS

Dimitrios Kalafatis (RN: 00351)

Supervisor: ***Fotios Kokkoras, Assistant Professor***

LARISSA, June 2024

Υπεύθυνη Δήλωση περί Ακαδημαϊκής Δεοντολογίας και Πνευματικών Δικαιωμάτων

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα πτυχιακή εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον, και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο Δηλών

(Υπογραφή)

Δημήτριος Καλαφάτης

20/6/2024

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων	**Κόκκορας Φώτιος,** Επίκουρος Καθηγητής, Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Κακαρόντζας Γεώργιος,** Καθηγητής, Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Κουτσονικόλα Βασιλική,** Ε.ΔΙ.Π., Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Θεσσαλίας

Ημερομηνία έγκρισης: 27-06-2024

Περίληψη

Το θέμα της πτυχιακής αφορά στην υλοποίηση ενός αλγορίθμου βαθιάς ενισχυτικής μάθησης σε ένα ανταγωνιστικό παίγνιο. Αρχικά μελετήθηκαν τα πεδία της ενισχυτικής μάθησης και της βαθιάς ενισχυτικής μάθησης, και ειδικότερα ο δρων πράκτορας, το περιβάλλον δραστηριοποίησής του, οι αμοιβές που αυτός λαμβάνει και σχετικοί αλγόριθμοι. Στη συνέχεια παρουσιάζεται το παίγνιο που επιλέχτηκε (Connect 4 - Σκορ 4) και γίνεται εκτενής περιγραφή της υλοποίησης του μηχανισμού εκπαίδευσης Deep Q-Learning στη γλώσσα προγραμματισμού Python, με χρήση βιβλιοθηκών μηχανικής μάθησης όπως η TensorFlow. Η τελική εφαρμογή έχει ικανοποιητική ικανότητα παιξίματος.

Abstract

This thesis deals with the implementation of a deep reinforcement learning algorithm in a competitive game. Initially, the fields of reinforcement learning, and deep reinforcement learning were studied, and in particular the active agent, its operating environment, the rewards/penalties he receives and related algorithms. Then the selected game (Connect 4) is presented and an extensive description of the implementation of the Deep Q-Learning training mechanism in the Python programming language is given, using machine learning libraries such as TensorFlow. The final application has satisfactory playability.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω πολύ την οικογένεια μου για την υποστήριξη που μου έδειξαν καθ' όλη τη διάρκεια της ακαδημαϊκής μου πορείας και θα ήθελα ιδιαίτερα να ευχαριστήσω όλο το ακαδημαϊκό προσωπικό του τμήματος για όλες τις σημαντικές γνώσεις που μου έδωσαν αυτά τα τέσσερα χρόνια των σπουδών μου.

Δημήτριος Καλαφάτης

20/6/2024

Περιεχόμενα

1 Εισαγωγή

Στόχος της πτυχιακής είναι η υλοποίηση ενός βαθιά ενισχυτικού αλγορίθμου σε ένα ανταγωνιστικό παίγνιο. Το παρόν κεφάλαιο ουσιαστικά αποτελεί μια προϊδέαση ως προς το περιεχόμενο του κάθε κεφαλαίου.

Στο δεύτερο κεφάλαιο παρουσιάζεται το πεδίο της (βαθιάς) ενισχυτικής μάθησης, δίνονται ορισμοί των βασικών στοιχείων της ενισχυτικής μάθησης (δηλ. πράκτορες, περιβάλλοντα, επιβραβεύσεις κ.λπ.), παρουσιάζονται οι θεωρητικές βάσεις από τις οποίες απορρέουν οι διαδικασίες ενισχυτικής μάθησης, παρουσιάζονται δύο αλγόριθμοι (Q-Learning και Deep Q-Learning, ο οποίος δεύτερος θα χρησιμοποιηθεί για την υλοποίηση) που χρησιμοποιούνται για την εκπαίδευση των μοντέλων και ένας αλγόριθμος (Epsilon-Greedy) που εξισορροπεί τις φάσεις μεταξύ εξερεύνησης (τυχαίες ενέργειες) και εκμετάλλευσης της μαθημένης πολιτικής. Επίσης στο τέλος του δεύτερου κεφαλαίου, θα δούμε δύο γνωστές υλοποιήσεις (η πρώτη υλοποίηση είναι στο τάβλι και η δεύτερη στο σκάκι) του πεδίου.

Στο τρίτο κεφάλαιο δίνεται μια περιγραφή του ανταγωνιστικού παίγνιου που αντιμετωπίζεται, αυτό του Σκορ 4. Παρουσιάζονται οι κανόνες του παιχνιδιού, η πολυπλοκότητα που διαθέτει και αν τελικά είναι ιδανικό για μια τέτοια υλοποίηση, επίσης γίνεται μια αναφορά στις βιβλιοθήκες της Python (δηλ. που συμβάλλει η κάθε μια) που θα χρησιμοποιηθούν ώστε να δημιουργήσουμε την υλοποίηση.

Στο τέταρτο κεφάλαιο στην αρχή δίνεται μια γενική περιγραφή της υλοποίησης (δηλ. μια επεξήγηση υψηλού επιπέδου) και μετά γίνεται μια πιο αναλυτική περιγραφή της υλοποίησης με παρουσίαση κάποιων τμημάτων του κώδικα. Ουσιαστικά αποτελεί την επεξήγηση του τρόπου με τον οποίο αντιμετωπίζεται το πρόβλημα.

Το πέμπτο κεφάλαιο αποτελεί τον οδηγό χρήσης για κάποιον που θέλει να τρέξει την υλοποίηση στο δικό του σύστημα (δίνονται οι εκδόσεις των βιβλιοθηκών που χρησιμοποιήθηκαν, συμβουλές για την παραμετροποίηση, τρόποι αντιμετώπισης θεμάτων κ.λπ.) επίσης στο κεφάλαιο αυτό δίνεται ένα παράδειγμα παιξίματος εναντίον του πράκτορα.

Τέλος, στο έκτο κεφάλαιο γίνεται μια γενική σύνοψη, τι πρέπει να προσέξει κάποιος που δημιουργεί μια παρόμοια υλοποίηση, το τι θα μπορούσε να αλλάξει κάποιος που θα ήθελε να

επεκτείνει την παρούσα υλοποίηση και στην διαφορά (αν υπάρχει) της διαδικασίας μάθησης μεταξύ των πρακτόρων βαθιάς ενισχυτικής μάθησης και των ανθρώπων.

2 Βασικά στοιχεία πεδίου και παρόμοιες υλοποιήσεις

Στην παρακάτω ενότητα θα δοθούν βασικοί ορισμοί της μηχανικής μάθησης για να κατανοήσει ο αναγνώστης τις διαφορές που έχει η ενισχυτική μάθηση με τις άλλες κατηγορίες άλλα και βασικά στοιχεία του πεδίου που ασχολείται η πτυχιακή δηλαδή της (βαθιάς) ενισχυτικής μάθησης, στην συνέχεια θα δούμε άλλες παρόμοιες δημοφιλής υλοποιήσεις που «τάραξαν» το πεδίο αυτό.

2.1 Μηχανική μάθηση

Όλοι οι υπολογιστικοί αλγόριθμοι που επιδεικνύουν ευφυή συμπεριφορά κατατάσσονται κάτω από την τεχνητή νοημοσύνη (AI), ωστόσο δεν είναι όλα τα συστήματα AI ικανά να μαθαίνουν. Η μηχανική μάθηση (ML) είναι ένας συγκεκριμένος τομέας εντός της AI που στοχεύει στη δημιουργία αλγορίθμων που μπορούν να μάθουν από δεδομένα για να επιλύσουν ευφυείς εργασίες. Η ML διαιρείται σε τρεις κύριες κατηγορίες: επιβλεπόμενη μάθηση, μη επιβλεπόμενη μάθηση και ενισχυτική μάθηση (Morales & Isbell, 2020). Αυτές οι κατηγορίες θα αναλυθούν παρακάτω.

Η επιβλεπόμενη μάθηση (SL) περιλαμβάνει την εκπαίδευση αλγορίθμων χρησιμοποιώντας δεδομένα που έχουν προηγουμένως επισημανθεί. Ένας άνθρωπος είναι υπεύθυνος για τη συλλογή και την επισήμανση των δεδομένων. Ο στόχος της SL είναι να δημιουργήσει μοντέλα που μπορούν να κάνουν ακριβείς προβλέψεις ή ταξινομήσεις σε νέα δεδομένα, με βάση τα μοτίβα που έχουν μάθει από τα επισημανθέντα δεδομένα. Ένα παράδειγμα της SL είναι μια εφαρμογή που αναγνωρίζει χειρόγραφους αριθμούς, όπου επισημανθείσες εικόνες χειρόγραφων ψηφίων χρησιμοποιούνται για να εκπαιδεύσουν ένα μοντέλο που μπορεί στη συνέχεια να αναγνωρίσει ψηφία σε νέες εικόνες (Morales & Isbell, 2020).

Η μη επιβλεπόμενη μάθηση (UL) ασχολείται με αλγορίθμους που μαθαίνουν από δεδομένα χωρίς προκαθορισμένες ετικέτες. Παρόλο που τα δεδομένα δεν απαιτούν επισήμανση, οι άνθρωποι παίζουν ακόμα ρόλο στον σχεδιασμό της διαδικασίας του πως συλλέγονται τα δεδομένα. Ο στόχος της UL είναι να μειώσει την πολυπλοκότητα των δεδομένων. Ένα παράδειγμα είναι η τμηματοποίηση πελατών σε διαφορετικές ομάδες με βάση τα δεδομένα τους,

το οποίο έχει ως αποτέλεσμα την αποκάλυψη μοτίβων και σχέσεων μεταξύ των πελατών (Morales & Isbell, 2020).

Η βαθιά μάθηση (DL) είναι μια τεχνική εντός της ML που χρησιμοποιεί βαθιά νευρωνικά δίκτυα για πολύπλοκη προσέγγιση συναρτήσεων. Η DL δεν είναι ένας ξεχωριστός κλάδος της ML αλλά μάλλον ένα σύνολο μεθόδων που χρησιμοποιούνται σε διάφορες εργασίες αυτής, συμπεριλαμβανομένων της επιβλεπόμενης, μη επιβλεπόμενης και ενισχυτικής μάθησης (Morales & Isbell, 2020).

2.2 Συστατικά ενισχυτικής μάθησης

Η ενισχυτική μάθηση (RL) είναι η διαδικασία όπου οι αλγόριθμοι μαθαίνουν πειραματιζόμενοι και μαθαίνοντας από τα αποτελέσματα των ενεργειών τους, χωρίς κανένα επισημανθέν δεδομένο ή προκαθορισμένες μεθόδους συλλογής δεδομένων. Ο κύριος στόχος είναι να αναπτυχθούν αλγόριθμοι που μπορούν να πάρουν αποφάσεις και να εκτελέσουν εργασίες αποτελεσματικά (Morales & Isbell, 2020).

Η βαθιά ενισχυτική μάθηση (DRL) είναι ένα υποσύνολο της τεχνητής νοημοσύνης που επικεντρώνεται στην ανάπτυξη αλγορίθμων υπολογιστών ικανών να αντιμετωπίζουν πολύπλοκες εργασίες που απαιτούν ευφυή συμπεριφορά. Οι αλγόριθμοι DRL χαρακτηρίζονται από την ικανότητά τους να μαθαίνουν από τις ενέργειές τους μέσα από μια διαδικασία δοκιμής και σφάλματος (trial and error), καθοδηγούμενη από ανατροφοδότηση που είναι τόσο διαδοχική όσο και αξιολογική (Morales & Isbell, 2020).

2.2.1 Πράκτορες

Στην DRL, ο όρος «πράκτορες» αναφέρεται σε συγκεκριμένα προγράμματα υπολογιστών. Αυτοί οι πράκτορες είναι αποκλειστικά υπεύθυνοι για τη λήψη αποφάσεων, χωρίς άλλες λειτουργίες. Για παράδειγμα, όταν εκπαιδεύουμε ένα ρομπότ για να χειρίζεται αντικείμενα, ο μηχανικός βραχίονας του ρομπότ δεν είναι ο πράκτορας· μόνο το λογισμικό λήψης αποφάσεων πληροί τα κριτήρια για να θεωρηθεί πράκτορας (Morales & Isbell, 2020).

Ο πράκτορας ακολουθεί έναν κύκλο τριών βημάτων: αλληλεπίδραση με το περιβάλλον, αξιολόγηση των ενεργειών του και βελτίωση των αποκρίσεών του (Morales & Isbell, 2020).

Οι πράκτορες κατηγοριοποιούνται με βάση το επίκεντρο της μάθησής τους: οι πράκτορες βασισμένοι σε πολιτικές μαθαίνουν πολιτικές, οι πράκτορες βασισμένοι σε τιμές μαθαίνουν συναρτήσεις τιμών, οι πράκτορες βασισμένοι σε μοντέλα μαθαίνουν μοντέλα, και οι πράκτορες τύπου actor-critic μαθαίνουν και πολιτικές και συναρτήσεις τιμών. Οι πράκτορες μπο-

ρούν να σχεδιαστούν για να ειδικεύονται σε έναν ή περισσότερους από αυτούς τους τομείς (Morales & Isbell, 2020).

Ο πράκτορας για να επιτύχει τους στόχους του μπορεί να χρησιμοποιήσει διάφορες στρατηγικές μηχανικής μάθησης, από δέντρα αποφάσεων και μηχανές διανυσματικής υποστήριξης (SVMs) έως νευρωνικά δίκτυα (NNs), για να προσεγγίσουν συναρτήσεις. Παρόλα αυτά, εστιάζουμε αποκλειστικά στα NNs, τα οποία αποτελούν την ουσία του «βαθύ» στην DRL. Είναι σημαντικό να αναγνωρίσουμε ότι τα νευρωνικά δίκτυα δεν είναι πάντα η βέλτιστη επιλογή για όλα τα προβλήματα· μιας και απαιτούν σημαντικές ποσότητες δεδομένων και μπορεί να είναι περίπλοκα για κατανόηση. Παρά ταύτα, τα NNs είναι μερικά από τα πιο αποτελεσματικά εργαλεία για την προσέγγιση συναρτήσεων, συχνά παρέχοντας ανώτερα αποτελέσματα. Τα τεχνητά νευρωνικά δίκτυα (ANNs) έχουν σχεδιαστεί για να μιμούνται τα βιολογικά νευρωνικά δίκτυα που βρίσκονται στον εγκέφαλο των ζώων, λειτουργώντας ως πολυεπίπεδοι μη-γραμμικοί προσεγγιστές. Αντί να είναι ένας συγκεκριμένος αλγόριθμος, ένα ANN είναι ένα πλαίσιο που εφαρμόζει μια σειρά μαθηματικών πράξεων σε διάφορα επίπεδα στα εισαγόμενα δεδομένα (Morales & Isbell, 2020).

2.2.2 Περιβάλλον

Το περιβάλλον περιλαμβάνει όλα όσα βρίσκονται εκτός του ελέγχου του πράκτορα, συμπεριλαμβανομένων όλων των εξωτερικών παραγόντων. Σκεπτόμενοι το ίδιο σενάριο με το ρομπότ, τα αντικείμενα που πρέπει να αναληφθούν, η επιφάνεια πάνω στην οποία ακουμπούν και οποιεσδήποτε εξωτερικές δυνάμεις όπως ο άνεμος, αποτελούν μέρος του περιβάλλοντος. Αυτό περιλαμβάνει και τον βραχίονα του ρομπότ, καθώς λειτουργεί με βάση τις αποφάσεις του πράκτορα αλλά υπόκειται σε απρόβλεπτους παράγοντες, καθιστώντας τον έτσι μέρος του περιβάλλοντος (Morales & Isbell, 2020).

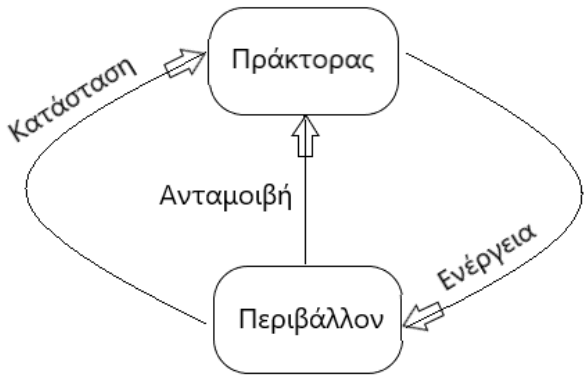
Το περιβάλλον χαρακτηρίζεται από μια συλλογή μεταβλητών που καθορίζουν το πρόβλημα που αντιμετωπίζεται. Στην περίπτωση του μηχανικού βραχίονα, η θέση και οι ταχύτητες κίνησης του βραχίονα είναι μεταβλητές που καθορίζουν το περιβάλλον. Αυτές οι μεταβλητές και οι πιθανές τους τιμές συνθέτουν αυτό που είναι γνωστό ως χώρος καταστάσεων. Ένα συγκεκριμένο σύνολο τιμών μεταβλητών σε οποιαδήποτε δεδομένη στιγμή ονομάζεται κατάσταση (Morales & Isbell, 2020).

Αξίζει να σημειωθεί ότι οι πράκτορες δεν έχουν πάντα πλήρη ορατότητα της πραγματικής κατάστασης του περιβάλλοντος. Λειτουργούν με βάση παρατηρήσεων, οι οποίες είναι μερικές απόψεις των καταστάσεων με βάση αυτό που μπορεί να ανιχνεύσει ο πράκτορας. Για

παράδειγμα, ο πράκτορας που ελέγχει τον μηχανικό βραχίονα μπορεί να έχει πρόσβαση μόνο σε οπτικά δεδομένα από κάμερες, όχι στις ακριβείς θέσεις των αντικειμένων (Morales & Isbell, 2020).

Το περιβάλλον οργανώνεται γύρω από μια συγκεκριμένη εργασία, ορισμένη από τη συνάρτηση επιβράβευσης. Αυτή η συνάρτηση παρέχει ανατροφοδότηση που είναι διαδοχική, αξιολογική και δειγματοληπτική. Για να επιτύχει τον στόχο του, ο πράκτορας πρέπει να εκδηλώσει νοημοσύνη ή γνωστικές δεξιότητες που συνδέονται με τη νοημοσύνη, όπως η στρατηγική σχεδίαση, η συλλογή πληροφοριών και η ικανότητα γενίκευσης (Morales & Isbell, 2020).

Ο πράκτορας και το περιβάλλον αλληλοεπιδρούνε μεταξύ τους πολλές φορές με τον χρόνο, η κάθε αλληλεπίδραση ονομάζεται χρονικό βήμα. Κατά τη διάρκεια ενός χρονικού βήματος, ο πράκτορας παρατηρεί, δρα και λαμβάνει νέες παρατηρήσεις και επιβραβεύσεις. Αυτά τα στοιχεία μαζί σχηματίζουν μια εμπειρία, η οποία είναι μια ευκαιρία για τον πράκτορα να μάθει και να βελτιώσει την απόδοσή του. Εάν μια εργασία που αντιμετωπίζει ένας πράκτορας έχει σαφές τέλος, τότε η εργασία του μπορεί να ονοματιστεί ως επεισοδιακή εργασία, εάν είναι συνεχής χωρίς να είναι καθορισμένο το τελικό σημείο, τότε ονομάζεται συνεχή εργασία. Η διαδικασία μάθησης εμπεριέχει πολλά χρονικά βήματα και επεισόδια, με τον πράκτορα συνεχώς να βελτιώνεται μέσω μιας διαδικασίας δοκιμής και σφάλματος. Πολλές από τις ενέργειες που πιθανώς να ληφθούν από τον πράκτορα θα μπορούσαν να εμπεριέχουν καθυστερημένες (δηλαδή εννοώντας ότι θα φανούν πιο μετά) επιδράσεις, και οι επιβραβεύσεις τους επίσης θα μπορούσαν να μην είναι άμεσες (Morales & Isbell, 2020).



Εικόνα 1: Γραφική αναπαράσταση αλληλεπίδρασης του πράκτορα με το περιβάλλον του.

2.2.3 Επιβραβεύσεις

Οι επιβραβεύσεις που δίνονται στον πράκτορα είναι συχνά λεπτές και δεν παρέχουν άμεση καθοδήγηση, δείχνοντας ένα θετικό αποτέλεσμα χωρίς να καθορίζουν τη σωστή ενέργεια.

Επομένως, ο πράκτορας πρέπει να μάθει από αξιολογική ανατροφοδότηση, η οποία απαιτεί μια ισορροπία μεταξύ της εξερεύνησης νέων πληροφοριών και της εκμετάλλευσης της υπάρχουσας γνώσης (Morales & Isbell, 2020).

Δεδομένου ότι ο πράκτορας λαμβάνει μόνο ένα δείγμα ανταμοιβών και δεν έχει πλήρη πρόσβαση στη συνάρτηση ανταμοιβής, και επειδή οι χώροι καταστάσεων και ενεργειών μπορούν να είναι τεράστιοι ή ακόμα και άπειροι, η μάθηση από περιορισμένη και μη άμεση ανατροφοδότηση είναι δύσκολη. Ο πράκτορας πρέπει να καταφέρει την γενίκευση (δηλαδή να ασκήσει την ικανότητα γενίκευσης του) από αυτά τα δείγματα ανατροφοδότησης που λαμβάνει (Morales & Isbell, 2020).

2.2.4 Ίχνος επιλεξιμότητας

Η μέθοδος κατανομής ανταμοιβής ή ποινής σε διάφορες καταστάσεις που συναντώνται σε προηγούμενες ενέργειες ονομάζεται ίχνος επιλεξιμότητας. Αυτή η έννοια περιλαμβάνει την παρακολούθηση του πώς οι ανταμοιβές ανατίθενται σε παρελθοντικά γεγονότα καθ' όλη τη διάρκεια μιας σειράς βημάτων. Η υλοποίηση και διαχείριση του πώς οι ανταμοιβές κατανέμονται αναλογικά σε παρελθοντικά βήματα χρησιμοποιώντας τα ίχνη επιλεξιμότητας μέσω μιας μεθόδου βάσει k περιγράφεται ως η προοπτική άποψη. Αντίθετα, η εναλλακτική μέθοδος, όπου καθυστερεί κανείς μέχρι να ολοκληρωθούν ορισμένα βήματα πριν αποδώσει αναδρομικά ανταμοιβές σε νωρίτερα βήματα, αναγνωρίζεται ως η αναδρομική άποψη της υλοποίησης των ιχνών επιλεξιμότητας (Sewak, 2019).

2.3 Θεωρητικές βάσεις

Για την πλήρη κατανόηση του πεδίου που μελετάμε, θα χρειαστεί να μελετήσομε τις θεωρητικές βάσεις από τις οποίες απορρέουν οι διαδικασίες της ενισχυτικής μάθησης, θα αναλυθούν οι βασικότερες παρακάτω.

2.3.1 Διαδικασία απόφασης Μάρκοφ

Η βάση οποιασδήποτε διαδικασίας RL είναι ριζωμένη στη διαδικασία απόφασης Μάρκοφ (MDP). Ο όρος Μάρκοφ προέρχεται από την ιδιότητα Μάρκοφ, η οποία είναι μια θεμελιώδης έννοια του φαινομένου της αλυσίδας Μάρκοφ και η MDP αντιπροσωπεύει μια παραλλαγή αυτού του φαινομένου. Η ιδιότητα Μάρκοφ υποδηλώνει ότι για μια διαδικασία η οποία έχει μεταβεί σε διάφορες καταστάσεις για να φτάσει σε μια τρέχουσα κατάσταση, αν αυτή

συμμορφώνεται με την ιδιότητα Μάρκοφ, τότε η υπό όρους κατανομή πιθανότητας της μελλοντικής κατάστασης καθορίζεται αποκλειστικά από την τρέχουσα κατάσταση, ανεξάρτητα από την πορεία που ακολούθησε η διαδικασία. Ως εκ τούτου, παρά την ύπαρξη πολλαπλών ακολουθιών για να φτάσει κανείς σε μια συγκεκριμένη κατάσταση, η υπό όρους κατανομή πιθανότητας για τις επόμενες καταστάσεις από εκείνη την κατάσταση θα παραμείνει σταθερή, ανεξάρτητα από την συγκεκριμένη ακολουθία που χρησιμοποίησε η διαδικασία για να φτάσει στην τρέχουσα κατάστασή της (Sewak, 2019).

Η Αλυσίδα Μάρκοφ χρησιμοποιεί την Ιδιότητα Μάρκοφ σε μια σειρά τυχαίων γεγονότων. Ορίζεται ως ένα τυχαίο μοντέλο που αποτελείται από μια σειρά γεγονότων όπου η πιθανότητα του επόμενου γεγονότος εξαρτάται εξ ολοκλήρου από το αποτέλεσμα του προηγούμενου γεγονότος. Σε μια Αλυσίδα Μάρκοφ, τα γεγονότα μπορούν να συμβούν κατά διακριτές ή συνεχείς περιόδους, αλλά πάντα περιλαμβάνουν έναν πεπερασμένο αριθμό δυνατών καταστάσεων (Sewak, 2019).

Η MDP περιγράφεται ως μια διαδικασία ελέγχου που λειτουργεί σε διακριτό χρόνο και ενσωματώνει μη ντετερμινιστικά (ή αλλιώς στοχαστικά) στοιχεία. Χρησιμοποιεί την αρχή της Αλυσίδας Μάρκοφ εντός ενός πλαισίου διαδικασίας απόφασης. Στο πεδίο της RL, η διαδικασία λήψης αποφάσεων είναι ουσιαστικά η πολιτική του πράκτορα, η οποία τον βοηθά στον εντοπισμό της βέλτιστης ενέργειας όταν βρίσκεται σε μια συγκεκριμένη κατάσταση (Sewak, 2019).

2.3.2 Προσομοίωση Μόντε Κάρλο

Η προσομοίωση Μόντε Κάρλο στοχεύει στην κατανόηση των υποκείμενων κατανομών των δεδομένων, συμπεριλαμβανομένων της συνδιακύμανσης μεταξύ των μεταβλητών, επιτρέποντας την παραγωγή νέων δεδομένων μέσω της δειγματοληψίας από αυτές τις παραγόμενες κατανομές. Αυτή η διαδικασία μιμείται την ανθρώπινη εξάρτηση από τις παρελθοντικές εμπειρίες για τη διαμόρφωση μορφωμένων εικασιών σχετικά με μελλοντικά αποτελέσματα (Sewak, 2019).

2.3.3 Μάθηση διαφορών χρόνου

Ενώ η μέθοδος προσομοίωσης Μόντε Κάρλο φαίνεται αποτελεσματική στη θεωρία, ένα σημαντικό της μειονέκτημα είναι η ανάγκη να περιμένει την ολοκλήρωση ενός μεγάλου μέρους επεισοδίων για να συσσωρεύσει τις απαραίτητες κατανομές (ή εμπειρίες) για να λειτουργήσει. Η μάθηση διαφορών χρόνου (TD) προσφέρει μια συμβιβαστική λύση μεταξύ των προσεγγίσεων δυναμικού προγραμματισμού, οι οποίες δεν απαιτούν προηγούμενη εμπειρία, και

των μεθόδων Μόντε Κάρλο, οι οποίες βασίζονται αποκλειστικά στην εμπειρία. Η $TD(0)$, μια παραλλαγή της TD, διαθέτει την ικανότητα να ενημερώνεται η μάθηση αυξητικά κατά πραγματικό χρόνο και χρησιμοποιεί μια μέθοδο δειγματοληψίας από δεδομένα με «αντικατάσταση» για την αναθεώρηση των στατιστικών του πληθυσμού, επιτρέποντας τη συνεχή ενημέρωση των τιμών ακόμη και κατά τη διάρκεια ενός επεισοδίου (Sewak, 2019).

Για την κατανόηση της σημασίας των προαναφερθέντων οφελών σε ένα πρακτικό σενάριο, ας σκεφτούμε τα πλεονεκτήματα της υπέρβασης των περιορισμών που σχετίζονται με την ανάγκη να περιμένουμε την ολοκλήρωση ενός επεισοδίου ώστε να ενημερωθεί η διαδικασία μάθησης. Η δυνατότητα ενημέρωσης τιμών χωρίς την ανάγκη αναμονής για την ολοκλήρωση ενός επεισοδίου υποδηλώνει ότι μια τέτοια μέθοδος θα μπορούσε να εφαρμοστεί σε καταστάσεις που περιλαμβάνουν επεισόδια μακράς διάρκειας (Sewak, 2019).

Ένα επιπλέον στοιχείο αυτού του περιορισμού περιλαμβάνει τα «συνεχή καθήκοντα», τα οποία ουσιαστικά είναι καθήκοντα που είναι μη επεισοδιακά, που σημαίνει ότι περιλαμβάνουν μια συνεχή διαδικασία χωρίς ένα ορισμένο τέλος. Εντός του επεισοδιακού πλαισίου, τα «συνεχή καθήκοντα» μπορούν να παρομοιαστούν ως ένα ατελείωτο μοναδικό επεισόδιο. Επιπλέον, υπάρχουν διαφορετικές εκδοχές της TD, συμπεριλαμβανομένου του $TD(1)$, και της ευρύτερης προσέγγισης που είναι γνωστή ως $TD(\lambda)$ (Sewak, 2019).

Η χρονική διαφορά λάμδα, ή $TD(\lambda)$, είναι μια ευρέως χρησιμοποιούμενη παραλλαγή της TD. Στην προσέγγιση $TD(0)$, η ενημέρωση τιμής για κάθε επανάληψη καθορίζεται προσθέτοντας την άμεση ανταμοιβή στην αξία της κατάστασης που ακολουθεί αμέσως μετά. Αυτή η μέθοδος υποθέτει ότι η διακύμανση στους στόχους τιμών για κάθε κατάσταση σε ένα συγκεκριμένο βήμα εξαρτάται αποκλειστικά από την κατάσταση του προηγούμενου βήματος στο "ταξίδι" του πράκτορα. Ωστόσο, αυτή η υπόθεση μπορεί να μην ισχύει καλά σε σενάρια όπου η ακολουθία παίζει κρίσιμο ρόλο. Η μέθοδος $TD(\lambda)$ αντιμετωπίζει και μετριάζει αυτόν τον περιορισμό που βρίσκεται στην προσέγγιση $TD(0)$ (Sewak, 2019).

Στην προσέγγιση $TD(\lambda)$, οι ανταμοιβές μπορούν να μοιραστούν σε προηγούμενες καταστάσεις που επισκέφθηκε διαδοχικά ο παράγοντας, με την υπόθεση ότι αυτές οι καταστάσεις έπαιξαν ρόλο στην απόκτηση της ανταμοιβής που λήφθηκε στο τρέχον βήμα κατά τον τρέχον κύκλο ενημέρωσης τιμής. Ο βαθμός στον οποίο οι ανταμοιβές αποδίδονται σε αυτές τις προηγούμενες καταστάσεις και ενέργειες καθορίζεται από την παράμετρο λ, η οποία κυμαίνεται στις τιμές από 0 έως 1 ($0 \leq \lambda \leq 1$). Αν οριστεί η παράμετρος λ στο 0 τότε συμφωνεί με τη μέθοδο $TD(0)$, επικεντρώνοντας αποκλειστικά στην άμεση ανταμοιβή από το πιο πρό-

σφατο βήμα. Αντιθέτως, μια τιμή λ του 1 κατανέμει ισόποσα την ανταμοιβή σε όλες τις προηγούμενες καταστάσεις και ενέργειες (Sewak, 2019).

Για οποιαδήποτε δεδομένη τιμή του λ, η κατανομή πίστωσης προσαρμόζεται από το λ^n, όπου το n αντιπροσωπεύει τον αριθμό των βημάτων πριν από το τελικό βήμα μέχρι τη στιγμή που μια κατάσταση προσπελάστηκε τελευταία φορά εκτελώντας την επιλεγμένη ενέργεια από την προηγούμενη κατάσταση. Αυτή η μέθοδος έχει εννοιολογικές ομοιότητες με μια στρατηγική όπου κάποιος ίσως καθυστερήσει τις ενημερώσεις για ένα συγκεκριμένο αριθμό βημάτων για να παρατηρήσει τη σειρά των καταστάσεων που διασχίζει ο πράκτορας. Σε ένα τέτοιο σενάριο, η απόφαση είναι να κατανεμηθούν ανταμοιβές μόνο για τις προηγούμενες καταστάσεις, ωστόσο με το $TD(\lambda)$, αυτό μπορεί να επιτευχθεί χωρίς την ανάγκη να παύσουμε για ενημερώσεις. Ως εκ τούτου, το $TD(\lambda)$ μπορεί να λειτουργήσει υπό υποθέσεις παρόμοιες με εκείνες της προσομοίωσης Μόντε Κάρλο (Sewak, 2019).

2.4 Αλγόριθμοι

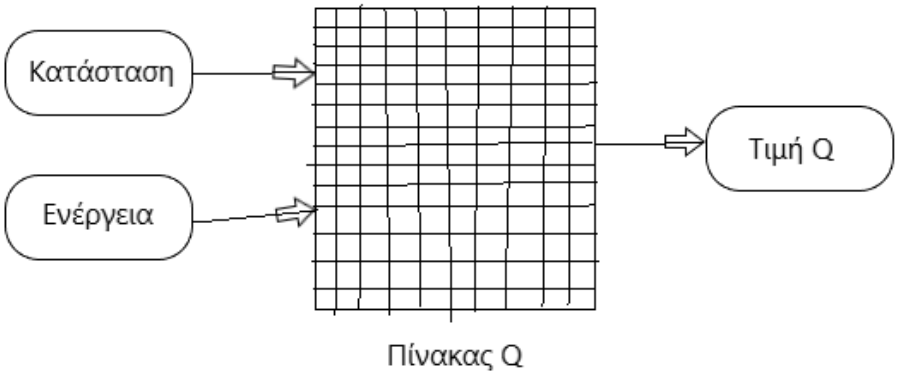
Παρακάτω θα εξηγηθεί το πως λειτουργεί ο αλγόριθμος Q-Learning που βοηθά με υπολογισμούς την επιλογή της βέλτιστης κίνησης όπως επίσης θα δούμε και την διαφορά που έχει με την «βαθιά» έκδοση του (Deep Q-Learning) και θα εξηγηθεί επίσης και ένας πολύ βασικός αλγόριθμος πολιτικής ο Epsilon-Greedy που στοχεύει στην εξισορρόπηση εξερεύνησης και εκμετάλλευσης.

2.4.1 Q-Learning

Ο αλγόριθμος Q-Learning χρησιμοποιεί την μάθηση διαφορών χρόνου για να αντιμετωπίσει το σκέλος της εκτίμησης σε ένα πρόβλημα. Ο αλγόριθμος με τον υπολογισμό της συνάρτησης τιμής-δράσης (συνάρτηση Q) βοηθά στην επιλογή της καλύτερης δράσης. Η συνάρτηση Q ενημερώνεται μετά από κάθε βήμα κατ' επανάληψη. Για την προσέγγιση αυτής της συνάρτησης χρησιμοποιεί την τωρινή τιμή Q, την νέα τιμή Q για την τότε κατάσταση και δράση, τον ρυθμό μάθησης και την ανταμοιβή που επέφερε η δράση στην κατάσταση (Sewak, 2019).

Ο μηχανισμός εξερεύνησης στον αλγόριθμο Q-Learning λειτουργεί ανεξάρτητα από την διαδικασία εκτίμησης, πράγμα που σημαίνει ότι ο αλγόριθμος δεν βασίζεται στην συνάρτηση Q για την επιλογή των ενεργειών του. Κατά την αρχήν αρχικοποιείται όλος ο πίνακας ή η μεταβλητή Q (ανάλογα την υλοποίηση) στο μηδέν και από μονός του αναπτύσσει μια αποτελεσματική πολιτική (Sewak, 2019).

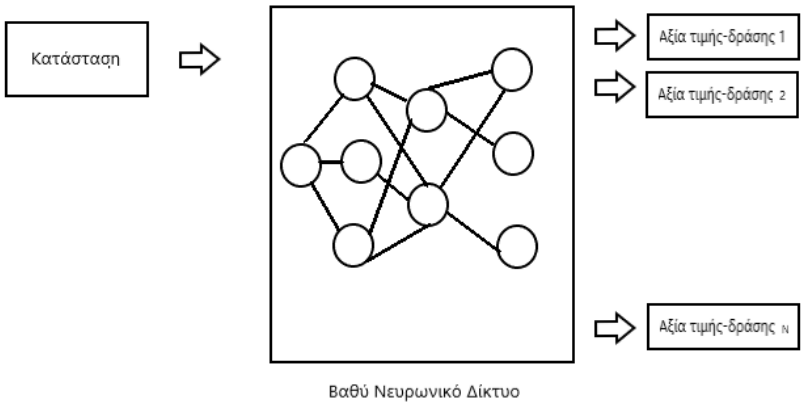
Άρα έχοντας υπόψη τα προαναφερόμενα μπορούμε να συμπεράνουμε πως ο αλγόριθμος απαιτεί μια στρατηγική για την διαχείριση την εξισορρόπηση της εξερεύνησης και της εκμετάλλευσης. Κατά την φάση της εξερεύνησης, μια δράση επιλέγεται τυχαία σύμφωνα με μια συνάρτηση πιθανότητας. Η επιλογή του να εξερευνήσει είτε να εκμεταλλευτεί γίνεται με τυχαίο τρόπο, και υπάρχουν αλγόριθμοι που υπολογίζουν αυτή την πιθανότητα της επιλογής των φάσεων, θα δούμε λίγο πιο κάτω τον αλγόριθμο Epsilon-Greedy (Sewak, 2019).



Εικόνα 2: Γραφική αναπαράσταση του αλγορίθμου Q-Learning.

2.4.2 Deep Q-Learning

Ο αλγόριθμος Deep Q-Learning αποτελεί έναν αλγόριθμο βαθιάς ενισχυτικής μάθησης, στον Q-Learning χρησιμοποιούμε έναν πίνακα Q που μας λέει την αναμενόμενη ανταμοιβή για την κάθε ενέργεια, στον Deep Q-Learning χρησιμοποιούμε ένα νευρωνικό δίκτυο για να μας βγάλει αυτή την αναμενόμενη ανταμοιβή, αυτή είναι η μεγαλύτερη διαφορά τους κατά τα άλλα ισχύουν ακόμα οι εξερευνήσεις και η μάθηση στο περιβάλλον όπως ακριβώς και στον αλγόριθμο Q-Learning. Η εφαρμογή του προτείνεται σε προβλήματα πιο πολύπλοκα (με εκατομμύρια χώρους καταστάσεων) όπου είναι αδύνατο ο απλός Q να κρατήσει τόσες πολλές πιθανότητες.



Εικόνα 3: Γραφική αναπαράσταση του αλγορίθμου Deep Q-Learning.

2.4.3 Epsilon-Greedy

Ο αλγόριθμος Epsilon-Greedy αποτελεί μια από τις πιο διαδεδομένες επιλογές για την εξισορρόπηση μεταξύ των φάσεων της εξερεύνησης και της εκμετάλλευσης. Χρησιμοποιεί μια σταθερά e, η οποία σηματοδοτεί την πιθανότητα της επιλογής της εξερεύνησης κατά τη διάρκεια κάθε επεισοδίου. Για παράδειγμα, εάν το e οριστεί στο 0.2, αυτό σημαίνει πως υπάρχει 20% πιθανότητα σε κάθε επεισόδιο να μπει στην φάση της εξερεύνησης και συνεπώς να επιλέξει μια τυχαία ενέργεια, ενώ αντιθέτως υπάρχει 80% πιθανότητα να μπει στην φάση εκμετάλλευσης που επιλέγει με βάση τις προβλέψεις που κάνει η συνάρτηση Q (παίρνει αυτή με την μεγαλύτερη αξία), οι οποίες ενημερώνονται μετά από κάθε επανάληψη (Sewak, 2019).

Όταν οριστεί μια τιμή e, παραμένει αμετάβλητη καθ' όλη την εκτέλεση. Ένα υψηλότερο e αυξάνει την πιθανότητα ο πράκτορας να ασχοληθεί με τυχαίες ενέργειες και συνεπώς να εξερευνήσει, ενώ ένα χαμηλότερο e αυξάνει την πιθανότητα την εκμετάλλευση της εκτιμωμένης τιμής Q. Η εύρεση της σωστής ισορροπίας μεταξύ είναι κρίσιμη για την ομαλή εκπαίδευση (Sewak, 2019).

2.5 Δημοφιλείς υλοποιήσεις

Παρακάτω θα γίνει μια παρουσίαση των υλοποιήσεων που ήρθαν και «ταρακούνησαν» το πεδίο της (βαθιάς) ενισχυτικής μάθησης και είναι ο λόγος που είναι πλέον τόσο δημοφιλής αυτό το πεδίο. Η πρώτη υλοποίηση που θα παρουσιαστεί είναι του Τεσαούρο, η δικιά του υλοποίηση είναι πάνω στο παιχνίδι του τάβλι, ενώ η δεύτερη είναι της ομάδας DeepMind και είναι πάνω στο σκάκι.

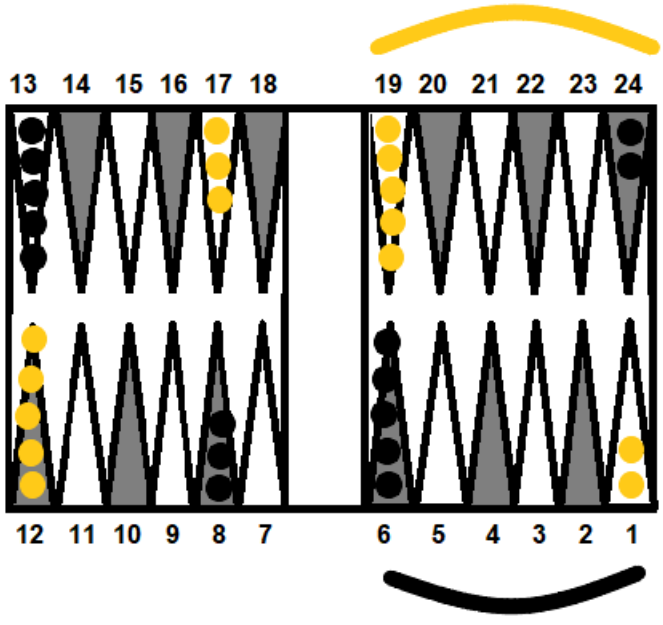
2.5.1 TD-Gammon

Οι πρώτες ενσωματώσεις τεχνητών νευρωνικών δικτύων σε προβλήματα ενισχυτικής μάθησης αρχίσαν να εφαρμόζονται τη δεκαετία του 1990. Ένα από τα πιο αξιοσημείωτα παραδείγματα είναι το TD-Gammon, ένα πρόγραμμα που σχεδιάστηκε για το παίξιμο τάβλι, αναπτύχθηκε από τον Τεσαούρο και τους συνεργάτες του. Αυτό το πρόγραμμα κατέκτησε το παιχνίδι αξιολογώντας τις θέσεις στον πίνακα χρησιμοποιώντας στρατηγικές ενισχυτικής μάθησης. Αν και οι μέθοδοι που χρησιμοποιήθηκαν στο TD-Gammon δεν μπορούν να καταταγούν αυστηρά ως βαθιά ενισχυτική μάθηση, αντιπροσωπεύει μία από τις πρώτες επιτυχείς

περιπτώσεις εφαρμογής τεχνητών νευρωνικών δικτύων στην αντιμετώπιση περίπλοκων προβλημάτων ενισχυτικής μάθησης (Morales & Isbell, 2020).

Κατά την δημιουργία του, απαιτούσε λίγες γνώσεις τάβλι για να ξεκινήσει, αλλά κατάφερε να επιτύχει κορυφαία ικανότητα παιξίματος που μπορούσε να συγκριθεί με τους κορυφαίους παίκτες τάβλι. Η επιτυχία του TD-Gammon οφειλόταν σε μια απλή ένταξη του αλγορίθμου TD(λ) με την προσέγγιση μη γραμμικής συνάρτησης, διευκολυνόμενη από ένα πολυεπίπεδο ANN που εκπαιδευόταν μέσω της αναδίπλωσης των σφαλμάτων TD (Sutton & Barto, 2018).

Το τάβλι είναι ένα από τα πλέον πιο αναγνωρίσιμα παιχνίδια στον κόσμο. Συνδυάζει όχι μόνο την στρατηγική αλλά και την τύχη. Έχει υποδειχθεί ότι ο αριθμός των επαγγελματιών παικτών τάβλι είναι μεγαλύτερο από εκείνο των επαγγελματιών παικτών του σκάκι. Το παιχνίδι χρησιμοποιεί 15 λευκά και 15 μαύρα πούλια, τα οποία τοποθετούνται σε μια σανίδα με 24 θέσεις, γνωστά και ως σημεία. Ας δούμε ένα παράδειγμα σε ένα αρχικό στάδιο του παιχνιδιού από την οπτική γωνία του λευκού παίκτη όπου ο λευκός έχει ρίξει ένα 5 και ένα 2 με τα ζάρια. Αυτό επιτρέπει στον παίκτη να μετακινήσει ένα πουλί 7 θέσεις συνολικά, είτε μετακινώντας ένα πουλί 5 θέσεις και ένα άλλο 2 θέσεις ή το ίδιο πουλί για τις συνολικές 7 θέσεις. Για παράδειγμα, η μετακίνηση ενός πουλί από το σημείο 12 στο σημείο 17 και ενός άλλου στο σημείο 14 είναι μια δυνατή κίνηση (Sutton & Barto, 2018).



Εικόνα 4: Αρχικό στάδιο τάβλι και απεικόνιση των σημείων.

Ο στόχος για τον λευκό παίκτη είναι να πάρει όλα τα πούλια στο τελευταίο τέταρτο της σανίδας (σημεία 19-24) και στη συνέχεια να τα βγάλει εντελώς από τη σανίδα. Η νίκη πηγαί-

νει στον παίκτη που αφαιρεί πρώτος όλα τα πούλια του από τη σανίδα. Μια μοναδική πτυχή του παιχνιδιού είναι η αλληλεπίδραση των πουλιών καθώς κινούνται σε αντίθετες κατευθύνσεις στη σανίδα. Τα αιχμαλωτισμένα πούλια τοποθετούνται προσωρινά στην μπάρα που βρίσκεται στο κέντρο της σανίδας, από όπου πρέπει να επανεισέλθουν στο παιχνίδι από την αρχή. Προστασία από το χτύπημα παρέχεται όταν δύο πούλια καταλαμβάνουν ένα μόνο σημείο, κάνοντας αδύνατο για τον αντίπαλο να πάει στο σημείο που είναι προστατευμένο. Κατά συνέπεια, ο λευκός δεν μπορεί να εφαρμόσει τη ρίψη 5-2 για να μετακινήσει πούλια από το 1 σημείο αν τα σημεία προορισμού καλύπτονται από δύο ή παραπάνω μαύρα πούλια. Η δημιουργία συνεχόμενων κατειλημμένων σημείων για να δημιουργηθεί ένα «μπλόκο» ενάντια στα πούλια του αντιπάλου αποτελεί μια από τις βασικές στρατηγικές μέσα στο παιχνίδι (Sutton & Barto, 2018).

Υπάρχουν αρκετές πολυπλοκότητες στο παιχνίδι του τάβλι πέραν των βασικών κανόνων που περιεγράφηκαν νωρίτερα. Διαθέτει 30 πούλια και 24 σημεία στον πίνακα, καθιστώντας εμφανές ότι ο αριθμός του χώρου καταστάσεων στο τάβλι είναι τεράστιο. Ο αριθμός των δυνατών κινήσεων από οποιαδήποτε δεδομένη θέση είναι σημαντικός, με έναν τυπικό ρόλο ζαριών να επιτρέπει έως και 20 διαφορετικά μονοπάτια που μπορεί να πάρει ο παίκτης. Η σχεδίαση εκ των προτέρων, απαιτεί να ληφθούν υπόψη και οι δυνατές κινήσεις του αντιπάλου, λαμβάνοντας υπόψη και τα διάφορα αποτελέσματα των ρίψεων από τα ζάρια του. Ως εκ τούτου, το δέντρο αποφάσεων του παιχνιδιού επεκτείνεται εκθετικά. Αυτή η πολυπλοκότητα που υπάρχει στο τάβλι καθιστά τις στρατηγικές ευρετικής αναζήτησης ανεφάρμοστες για το τάβλι ακόμα και εάν έχουν αποδειχθεί επιτυχημένες σε άλλα παιχνίδια (Sutton & Barto, 2018).

Τα χαρακτηριστικά του παιχνιδιού ταιριάζουν γάντι με τα δυνατά σημεία των τεχνικών της διαφόρους μάθησης. Παρά το σημαντικό στοιχείο της τυχαίοτητας που υπάρχει στο παιχνίδι, μια πλήρης περιγραφή της κατάστασης είναι πάντα προσβάσιμη. Προχωρά μέσα από μια σειρά από ενέργειες και καταστάσεις, καταλήγοντας σε μια νίκη για έναν από τους δύο παίκτες, η οποία κλείνει τη συνεδρία. Αυτό το αποτέλεσμα μπορεί να θεωρηθεί ως η τελική ανταμοιβή που χρειάζεται να προβλεφθεί. Ωστόσο, οι θεωρητικές έννοιες που συζητήθηκαν μέχρι αυτό το σημείο δεν είναι αποτελεσματικές για αυτή την πρόκληση. Ο τεράστιος όγκος των δυνατών καταστάσεων καθιστά ανεφάρμοστο έναν πίνακα αναζήτησης και η παρουσία ενός αντιπάλου εισάγει απροβλεψιμότητα και δυναμικές αλλαγές με την πάροδο του χρόνου (Sutton & Barto, 2018)

Η υλοποίηση του Τεσαούρο χρησιμοποίησε μια μη γραμμική παραλλαγή του TD(λ) για την εκτίμηση της πιθανότητας νίκης από οποιαδήποτε δεδομένη θέση στον πίνακα, που συμβολίζεται ως v'(s, w), όπου το s αντιπροσωπεύει την κατάσταση ή την θέση στον πίνακα. Το σύστημα σχεδιάστηκε για να προσεγγίσει την πιθανότητα νίκης από την κατάσταση s, με τα έπαθλα να ορίζονται στο μηδέν σε όλες τις στιγμές εκτός όταν συνέβαινε μια νίκη. Για τον υπολογισμό της συνάρτησης αξίας, το TD-Gammon χρησιμοποίησε ένα ANN. Το πραγματικό δίκτυο περιλάμβανε δύο επιπλέον μονάδες στο τελικό του επίπεδο, σχεδιασμένες ειδικά για να υπολογίζουν την πιθανότητα νίκης για κάθε παίκτη. Η αρχιτεκτονική του δικτύου διέθετε ένα εισαγωγικό επίπεδο που αντιπροσώπευε τη θέση του πίνακα τάβλι, ένα κρυφό επίπεδο και μια τερματική μονάδα εξόδου που παρείχε μια αξιολόγηση της αξίας της θέσης (Sutton & Barto, 2018).

Το δίκτυο στην αρχική έκδοση του TD-Gammon δεχόταν είσοδο μέσω 198 μονάδων. Για κάθε σημείο στον πίνακα τάβλι, υπήρχαν τέσσερις μονάδες αφιερωμένες στην αναπαράσταση του αριθμού των λευκών πιονιών που υπήρχαν. Αν ένα σημείο δεν είχε λευκά πιόνια, αυτές οι τέσσερις μονάδες θα ήταν όλες μηδέν. Ένα μόνο πιόνι σε ένα σημείο θα έθετε την πρώτη μονάδα σε 1, συμβολίζοντας την παρουσία ενός blot δηλαδή ενός πιονιού που είναι ευάλωτο στο να χτυπηθεί από τον αντίπαλο και να αιχμαλωτιστεί στην μπάρα. Δύο ή περισσότερα πιόνια θα ενεργοποιούσαν τη δεύτερη μονάδα σε 1, απεικονίζοντας ένα made point που ο αντίπαλος δεν μπορεί να ούτε να χτυπήσει τον αντίπαλο ούτε να βάλει πιόνι. Ακριβώς τρία πιόνια θα άναβαν την τρίτη μονάδα σε 1, σημαίνοντας ένα single spare, που σημαίνει ότι υπάρχει ένα επιπλέον πιόνι εκτός από τα δύο που εξασφαλίζουν το σημείο. Περισσότερα από τρία πιόνια θα προσάρμοζαν την τιμή της τέταρτης μονάδας με βάση τον τύπο (n-3)/2, όπου n είναι ο συνολικός αριθμός των πιονιών, για να αναπαρασταθούν πολλαπλά spares με γραμμικό τρόπο (Sutton & Barto, 2018).

Για κάθε ένα από τα 24 σημεία στον πίνακα, αποδόθηκαν τέσσερις μονάδες για τα λευκά πιόνια και τέσσερις για τα μαύρα πιόνια, οδηγώντας σε ένα σύνολο 192 μονάδων. Επιπλέον, δύο μονάδες χρησιμοποιήθηκαν για να αντιπροσωπεύσουν τον αριθμό των λευκών και μαύρων πιονιών στην μπάρα, υπολογιζόμενος ως n/2, όπου n αντιπροσωπεύει τα συνολικά πιόνια στην μπάρα. Άλλες δύο μονάδες αφιερώθηκαν για να δείξουν τον αριθμό των λευκών και μαύρων πιονιών που είχαν αφαιρεθεί επιτυχώς από το παιχνίδι, χρησιμοποιώντας τον τύπο n/15, όπου n είναι τα συνολικά πιόνια που έχουν απομακρυνθεί. Δύο τελικές μονάδες χρησιμοποιήθηκαν για να δηλώσουν ποιανού είναι η σειρά, λευκού ή μαύρου. Η απόφαση στο πως θα αναπαρίστανται η κατάσταση του παιχνιδιού, πάρθηκε με την σκέψη ότι είναι απλή και

ελαχιστοποιούσε τον συνολικό αριθμό των χρησιμοποιούμενων μονάδων (Sutton & Barto, 2018).

Για την υλοποίηση της διαδικασίας μάθησης ήταν απαραίτητο να υπάρχει μια συνεχής ροή αγώνων τάβλι, αυτό επιτεύχθηκε με μια μορφή αυτόμαθησης όπου το πρόγραμμα ανταγωνιζόταν με τον εαυτό του (έπαιζε και από τις δύο μεριές). Για την απόφαση κάθε κίνησης του, το TD-Gammon αξιολογούσε περίπου 20 πιθανές κινήσεις για κάθε ρίψη ζαριών και τις επακόλουθες διατάξεις του πίνακα που αυτές οι κινήσεις θα δημιουργούσαν, γνώστες ως μετακαταστάσεις. Το σύστημα στη συνέχεια αξιολογούσε την αξία κάθε πιθανού αποτελέσματος. Η στρατηγική περιλάμβανε την επιλογή της κίνησης που προβλεπόταν ότι θα οδηγούσε στην πιο ευνοϊκή διάταξη του πίνακα. Κάθε αγώνας θεωρούνταν ένα επεισόδιο, με τη σειρά των διατάξεων του πίνακα να σχηματίζει τις καταστάσεις, σημειωμένες ως S0, S1, S2 και ούτω καθεξής. Ο Τεσαούρο χρησιμοποίησε τον μη γραμμικό κανόνα διαφόρου μάθησης με έναν εναλλακτικό τρόπο, εφαρμόζοντάς τον μετά από κάθε μεμονωμένη κίνηση (Sutton & Barto, 2018).

Τα αρχικά βαρύ του δικτύου ανατέθηκαν σε τυχαίους αριθμούς, κάνοντας έτσι τις αρχικές εκτιμήσεις εντελώς τυχαίες. Ως αποτέλεσμα, τα πρώτα λίγα παιχνίδια επειδή βασιζόντουσαν σε τυχαίες εκτιμήσεις (με αποτέλεσμα κακών κινήσεων) παρατείνονταν για χιλιάδες κινήσεις πριν αναδειχθεί ένας νικητής, περισσότερο από τύχη παρά από δεξιότητα. Ωστόσο, μετά από το παίξιμο αρκετών παιχνιδιών, υπήρξε μια σημαντική και γρήγορη βελτίωση στην απόδοση (Sutton & Barto, 2018).

Μετά από ένα σύνολο 300.000 αγώνων αυτοπαιχνιδιού, η αρχική έκδοση του TD-Gammon (0.0) επέτυχε ένα επίπεδο δεξιότητας συγκρίσιμο με αυτό του πιο προχωρημένου προγράμματος που ήταν διαθέσιμο εκείνη την εποχή. Αυτό το αποτέλεσμα ήταν ιδιαίτερα αξιοσημείωτο δεδομένου ότι τα υπόλοιπα προγράμματα για το τάβλι βασιζόντουσαν σε μεγάλο βαθμό από λεπτομερείς γνώσεις του παιχνιδιού. Η ικανότητα του να ταιριάζει ή να υπερβαίνει την απόδοση από άλλα παρόμοια προγράμματα αποτελεί μια επιβεβαίωση της αποτελεσματικότητας που προσφέρει η μάθηση μέσω του αυτοπαιχνιδιού (Sutton & Barto, 2018).

Παρά την έλλειψη προηγούμενης γνώσης στρατηγικών ειδικών για το τάβλι, η επίδοση της αρχικής εκδόσης του TD-Gammon εντυπωσίαζε. Στις επόμενες εκδόσεις του TD-Gammon (3.0 και 3.1), ενσωματώθηκαν και γνώσεις στρατηγικών διατηρώντας την προσέγγιση αυτοπαιχνιδιού της μάθησης διαφοράς χρόνου. Στις νέες εκδόσεις ενσωματώθηκαν 160 κρυφές μονάδες και χρησιμοποιούσαν μια αναζήτηση που μπορούσε να βλέπει έως και τρεις

κινήσεις μπροστά. Το TD-Gammon αποτέλει παράδειγμα πως ο συνδυασμός των μαθημένων συναρτήσεων αξίας με τις αναζητήσεις λήψης αποφάσεων, παρόμοιες με τις ευρετικές αναζητήσεις και τις τεχνικές αναζήτησης δένδρου Μόντε Κάρλο, μπορεί να ενισχύσει το στρατηγικό παιχνίδι. Ο χρόνος που έπαιρνε για να κάνει μια κίνηση (λήψη απόφασης) ήταν μεταξύ 5 έως 10 δευτερολέπτων (Sutton & Barto, 2018).

2.5.2 AlphaGo Zero

Για την δημιουργία του AlphaGo Zero η ομάδα της DeepMind βασίστηκε στις παρατηρήσεις που απέκτησε από το προηγούμενο δημιούργημα της το AlphaGo, όπως απορρέει και από το όνομα του, το AlphaGo Zero χρησιμοποιούσε μηδέν δεδομένα από ανθρώπινα παιχνίδια παρά μόνο γνώριζε τους κανόνες του παιχνιδιού Go. Η μάθηση του βασιζόταν αποκλειστικά στην ενίσχυση του μέσω του αυτοπαιχνιδιού. Υλοποιήθηκε μια εναλλασσόμενη στρατηγική μεταξύ της αξιολόγησης και της βελτίωσης των πολιτικών του, χρησιμοποιούσε επίσης κατά τη φάση μάθησης την αναζήτηση δέντρου Μόντε Κάρλο (Sutton & Barto, 2018).

Η αναζήτηση δέντρου Μόντε Κάρλο που χρησιμοποιήθηκε στο AlphaGo Zero ήταν πιο απλή σε σύγκριση με την έκδοση που χρησιμοποιήθηκε από τον προκάτοχο του λόγω το ότι μετά από κάθε αναζήτηση δεν πραγματοποιούσε προσομοίωση ενός παιχνιδιού μέχρι την τελική έκβαση του παρά μόνο σε έναν φύλλο κόμβο εντός του ήδη υπάρχοντος δέντρου. Παρομοίως και στα δύο, οι επαναλήψεις MCTS ενημερώνονταν από τις προβλέψεις ενός βαθέως συνελεκτικού δικτύου, αυτό το δίκτυο παίρνει ως είσοδο την ακατέργαστη μορφή του πίνακα και παράγει δύο εξόδους: μια τιμή, v, που προβλέπει την πιθανότητα νίκης με βάση την τρέχων κατάσταση του στον πίνακα, και ένα διάνυσμα, p, που δείχνει τις πιθανότητες όλων των δυνατών κινήσεων (Sutton & Barto, 2018).

Η αρχιτεκτονική του δικτύου χαρακτηριζόταν από έναν «διπλοκέφαλο» σχεδιασμό, κάθε κεφάλι οδηγούσε σε δικό του σύνολο μονάδων εξόδου. Ειδικότερα, το πρώτο κεφάλι συνδεόταν με 362 μονάδες εξόδου, παράγοντας 193 πιθανότητες κινήσεων p, λαμβάνοντας υπόψη κάθε δυνατή τοποθέτηση ενός πετραδιού συν την επιλογή να προσπεράσει αυτό τον γύρο. Το δεύτερο κεφάλι συνδεόταν με μια μονάδα εξόδου, που παρείχε μια τιμή v, αντιπροσωπεύοντας την εκτιμώμενη πιθανότητα νίκης με βάση της δεδομένης κατάστασης του πίνακα. Το δίκτυο αποτελούνταν από 41 συνελεκτικά επίπεδα, τα οποία ενισχύονταν με κανονικοποίηση (Sutton & Barto, 2018).

Η τελική αξιολόγηση από την μεριά των ανθρώπων στον αλγόριθμο του AlphaGo Zero, έγινε με την χρήση ενός μεγαλύτερου τεχνητού νευρωνικού δικτύου και την εκπαίδευση του

μέσω 29 εκατομμυρίων παιχνιδιών εναντίον του για μια περίοδο περίπου 40 ημερών. Αυτή η εκδοχή έφτασε σε ένα υψηλότερο σκορ από ότι έφτασε το πιο προηγμένο για εκείνη την εποχή AlphaGo Master, το όποιο σε αντίθεση με το AlphaGo Zero είχε ενισχυθεί με δεδομένα και στρατηγικές. Σε μια σειρά από 100 παιχνίδια, το ενισχυμένο AlphaGo Zero νίκησε το AlphaGo Master κερδίζοντας 89 έναντι 11 (Sutton & Barto, 2018).

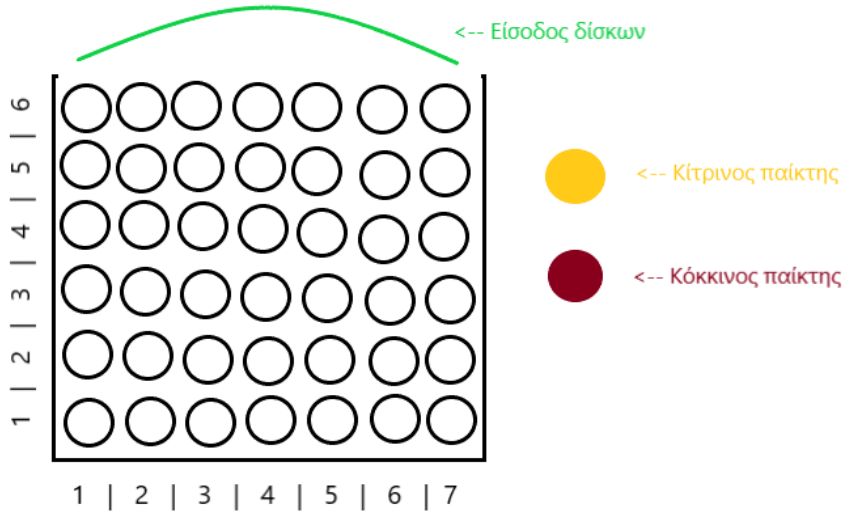
Το AlphaGo Zero είναι μια απόδειξη ότι η βαθιά ενισχυτική μάθηση, με μια απλοποιημένη μορφή MCTS είναι ικανή για μια πολύ υψηλή απόδοση σε οποιοδήποτε παιχνίδι χωρίς να εξαρτάται από δεδομένα ανθρώπων (Sutton & Barto, 2018).

3 Περιγραφή προβλήματος και εργαλείων

Στην παρακάτω ενότητα γίνεται μια εισαγωγή στο παιχνίδι Σκορ 4, αναφέρονται οι κανόνες του παιχνιδιού και οι στόχοι για την επίτευξη της νίκης, οι στρατηγικές που απαιτούνται από την φύση του παιχνιδιού, η ανταγωνιστική φύση του, η πολυπλοκότητα του χώρου καταστάσεων, το στρατηγικό του βάθος και γιατί είναι καλή επιλογή για την υλοποίηση ενός αλγορίθμου βαθιάς ενισχυτικής μάθησης. Επίσης θα γίνει αναφορά στα εργαλεία που θα χρησιμοποιηθούν.

3.1 Σκορ 4

Το Σκορ 4 είναι ένα κλασικό παιχνίδι δύο παικτών που περιλαμβάνει τοποθέτηση δίσκων σε κατακόρυφο πλέγμα. Το πλέγμα αποτελείται από 6 σειρές και 7 στήλες, οι παίκτες έχουν χρωματιστούς δίσκους, συνήθως κίτρινους και κόκκινους. Ο στόχος είναι απλός αλλά βαθιά στρατηγικός: να είσαι ο πρώτος που θα σχηματίσει μια οριζόντια, κάθετη ή διαγώνια γραμμή από τέσσερις δίσκους του ίδιου χρώματος.



Εικόνα 5: Ταμπλό του Σκορ 4.

Οι παίκτες εναλλάσσονται ρίχνοντας έναν από τους χρωματιστούς δίσκους τους από την κορυφή σε οποιαδήποτε από τις επτά στήλες. Ο δίσκος καταλαμβάνει τον χαμηλότερο διαθέσιμο χώρο μέσα στη στήλη. Το παιχνίδι τελειώνει όταν ένας παίκτης σχηματίσει μια

γραμμή από τέσσερις δίσκους του σε μια σειρά ή αν το πλέγμα γεμίσει πλήρως χωρίς να έχουν πληρωθεί οι όροι νίκης με αποτέλεσμα να λήξει ισόπαλο.

Το παιχνίδι απαιτεί υψηλό επίπεδο τακτικής λήψης αποφάσεων και προγραμματισμού εκ των προτέρων. Οι παίκτες πρέπει να επικεντρώνονται όχι μόνο στη δημιουργία μιας γραμμής τεσσάρων αλλά και στο να μπλοκάρουν στρατηγικά τις προσπάθειες των αντιπάλων τους. Οι διπλές διατάξεις, όπου μια γραμμή τριών δίσκων μπορεί να ολοκληρωθεί από δύο κατευθύνσεις, είναι ιδιαίτερα ισχυρές. Οι προχωρημένοι παίκτες μπορούν και σχεδιάζουν αρκετές κινήσεις μπροστά, δημιουργώντας πολλαπλές πιθανές γραμμές τεσσάρων ενώ ταυτόχρονα μπλοκάρουν τον αντίπαλο.

3.1.1 Ανταγωνιστικότητα και πολυπλοκότητα

Το Σκορ 4 είναι ανταγωνιστικό λόγω της απαίτησης για τους παίκτες να προσαρμόζουν συνεχώς τις στρατηγικές τους με βάση την εξελισσόμενη κατάσταση του πίνακα. Η ανάγκη να προβλέπουν τις κινήσεις του αντιπάλου και να τις αντικρούσουν προσθέτει ένα επίπεδο πολυπλοκότητας, κάνοντας κάθε αγώνα μοναδικό. Το παιχνίδι δεν αφορά μόνο την άμεση κίνηση αλλά και την προετοιμασία για μελλοντική επιτυχία ή την πρόληψη της νίκης του αντιπάλου.

Το ταμπλό του Σκορ 4 διαθέτει 7 στήλες και 6 σειρές, αυτό οδηγεί σε έναν τεράστιο αριθμό πιθανών καταστάσεων παιχνιδιού. Με την έναρξη κάθε νέου γύρου, η πολυπλοκότητα αυξάνεται εκθετικά λόγω όλων των δυνατών διακλαδώσεων που δημιουργούνται. Μια κίνηση του παίκτη αλλάζει την κατάσταση του παιχνιδιού, αναγκάζοντας και τους δύο παίκτες να αξιολογούν συνεχώς το ταμπλό και να προσαρμόζουν τις στρατηγικές τους. Έχει υπολογιστεί ότι το Σκορ 4 έχει περίπου τεσσεράμισι τρισεκατομμύρια πιθανές χωρικές καταστάσεις.

Το στρατηγικό βάθος του Σκορ 4 προκύπτει από το ότι η μελλοντική κατάσταση του παιχνιδιού επηρεάζεται με κάθε μια κίνηση. Οι παίκτες θα πρέπει να λαμβάνουν υπόψη όχι μόνο την άμεση επίδραση των ενεργειών τους αλλά και το πως να προετοιμάσουν το «έδαφος» για τις επόμενες κινήσεις. Αυτή η προνοητικότητα απαιτεί κατανόηση των δυνατότητων που εμφανίζονται από τις διακλαδιζόμενες καταστάσεις του παιχνιδιού και την ικανότητα πρόβλεψης και επηρεασμού της απόφασης του αντιπάλου.

3.1.2 Ιδανικότητα για την υλοποίηση

Η ιδανικότητα του Σκορ 4 για την εφαρμογή αλγορίθμων βαθιάς ενισχυτικής μάθησης απορεέι λόγω του δομημένου περιβάλλοντος και των σαφών κανόνων του. Το παιχνίδι

παρέχει ένα τέλειο σενάριο για την εκπαίδευση πρακτόρων, κάθως περιλαμβάνει έναν πεπερασμένο αλλά τεράστιο χώρο παιχνιδιού και απαιτεί εκτός από στρατηγικό βάθος και προσαρμοστικότητα. Μέσω του επαναλαμβανόμενου παιξίματος, οι πράκτορες μπορούν να ανακαλύψουν περίπλοκα μοτίβα παίζοντας πολλαπλά παιχνίδια με τον εαυτό τους ή με κάποιο προκαθορισμένο σετ. Αυτή η διαδικασία επιτρέπει στον πράκτορα να μάθει τις βέλτιστες κινήσεις σε διάφορες καταστάσεις, βελτιώνοντας το παιχνιδί και την στρατηγική για την κάθε απόφαση που παίρνει.

Είναι τέτοια η φύση του παιχνιδιού που σε συνδυασμό με την πολυπλοκότητα που διαθέτει και το απαιτούμενο του προγραμματισμού αρκετών κινήσεων μπροστά, το καθιστά ένα εξαιρετικό περιβάλλον για την εξερεύνηση τεχνικών τεχνητής νοημοσύνης. Οι πράκτορες μπορούν να αναπτύξουν μια λεπτομερή κατανόηση των στρατηγικών του παιχνιδιού, οι οποίες στρατηγικές θα μπορούσαν να εφαρμοστούν και σε άλλα σενάρια επίλυσης προβλημάτων πέρα από το ίδιο το παιχνίδι.

Συμπεραίνουμε ότι, ο συνδυασμός των απλών κανόνων και της βαθιάς στρατηγικής πολυπλοκότητας που διαθέτει το Σκορ 4, το καθιστά μια «αρένα δοκιμασίας» για την ευφυΐα του πράκτορα και συνεπώς το καθιστά κατάλληλο για υλοποίηση ενός αλγορίθμου βαθιάς ενισχυτικής μάθησης ή και άλλων παρόμοιων τεχνικών τεχνητής νοημοσύνης.

3.2 Βιβλιοθήκες

Για την υλοποίηση ενός αλγορίθμου βαθιάς ενισχυτικής μάθησης για το Σκορ 4 θα χρειαστεί να αξιοποιήσουμε τις δυνατότητες αρκετών ισχυρών βιβλιοθηκών, καθεμία από αυτές θα συνεισφέρει μοναδικά. Οι βιβλιοθήκες που θα χρειαστούμε είναι η TensorFlow, η NumPy και η Jax. Κάθε μια από αυτές τις βιβλιοθήκες παρέχει λειτουργείες που τις καθιστούν απαραίτητες για διάφορες πτυχές της δημιουργίας ενός μοντέλου. Θα δούμε αργότερα στο επόμενο κεφάλαιο πως κάθε μια από αυτές εντάσσεται στην ανάπτυξη ενός μοντέλου DRL για το Σκορ 4.

3.2.1 TensorFlow

Το TensorFlow αποτελεί μια βιβλιοθήκη ανοικτού κώδικα, η οποία χρησιμοποιείται για να καταστήσει τη μηχανική μάθηση πιο εύκολη και πιο γρήγορη. Είναι ιδιαίτερα κατάλληλη για την δημιουργία και εκπαίδευση απλών και βαθύ νευρωνικών δικτύων, τα όποια αποτελούν την καρδία των αλγόριθμών DRL. Η βιβλιοθήκη αυτή προσφέρει διεπαφές (APIs) υψηλού επιπέδου όπως το Keras, καθιστώντας την δημιουργία νευρωνικών δικτυών απλή διαδικασία,

προσφέρει και APIs χαμηλού επιπέδου για την δυνατότητα παρέμβασης πάνω στην αρχιτεκτονική του μοντέλου και στην διαδικασία της εκπαίδευσης του.

3.2.2 NumPy

Η βιβλιοθήκη NumPy αποτελεί ένα βασικό εργαλείο για την επιστημονική υπολογιστική στην Python. Παρέχει μαθηματικές συναρτήσεις για πράξεις και επεξεργασίες πάνω σε μικρούς, μεγάλους και πολυδιάστατους πίνακες.

3.2.3 JAX

Η JAX είναι μια βιβλιοθήκη που επεκτείνει ουσιαστικά την NumPy προσθέτοντας αυτόματη διαφοροποίηση και την υποστήριξη GPU, καθιστώντας την βιβλιοθήκη υψηλών επιδόσεων για την ανάπτυξη περίπλοκων μοντέλων μηχανικής μάθησης.

4 Υλοποίηση DQL στο Σκορ 4

Σε αυτή την ενότητα, θα δοθεί πρώτα μια γενική περιγραφή της υλοποίησης και μετέπειτα θα γίνει μια αναλυτική περιγραφή αυτών των σημείων που είναι πιο σημαντικά (και ίσως πιο περίπλοκα για την κατανόηση τους) για την εύρυθμη λειτουργία της υλοποίησης.

4.1 Γενική περιγραφή

Η υλοποίηση ενός αλγορίθμου βαθιάς ενισχυτικής μάθησης για το Σκορ 4, στην προκειμένη περίπτωση του αλγορίθμου Deep Q-Learning πρέπει να περιλαμβάνει αρκετά στοιχεία για να μπορεί να μαθαίνει και να λαμβάνει αποφάσεις αποτελεσματικά. Σε αυτή την υποενότητα θα γίνει μια γενική περιγραφή της υλοποίησης δηλαδή μια επισκόπηση υψηλού επιπέδου, εστιάζοντας στα βασικά στοιχεία της υλοποίησης, τις εισόδους και τις εξόδους του δικτύου μαζί με τον κύριο στόχο του πράκτορα και τα σημαντικά βήματα που γίνονται για την επίλυση του παιχνιδιού.

Όσον αφορά το πρώτο βασικό στοιχείο, που είναι η αρχιτεκτονική του νευρωνικού δικτύου, η κλάση NeuralNetworkAgent είναι υπεύθυνη για τον ορισμό της δομής του μοντέλου του νευρωνικού δικτύου. Το δίκτυο αυτό αποτελείται από στρώματα συνέλιξης ακολουθούμενα από πυκνά στρώματα, με ειδικές συναρτήσεις ενεργοποίησης για κάθε στρώμα. Το δίκτυο αρχικοποιείται στην αρχή με τυχαία βάρη, η είσοδος που παίρνει το νευρωνικό δίκτυο είναι η προεπεξεργασμένη κατάσταση του παιχνιδιού, ενώ στην έξοδο του νευρωνικού παράγονται οι τιμές Q για κάθε δυνατή ενέργεια (οι ενέργειες αναπαρίστανται ως διανύσματα με κωδικοποίηση τύπου one-hot) δεδομένης της τρέχουσας κατάστασης του παιχνιδιού. Τις περισσότερες φορές επιλέγεται η ενέργεια με την υψηλότερη τιμή Q όμως υπάρχει και κάποια πιθανότητα επιλογής μιας τυχαίας ενέργειας βάσει την στρατηγική εξερεύνησης (θα αναφερθούμε σε αυτό πιο κάτω).

Η κλάση ExperienceReplayBuffer είναι υπεύθυνη για την αποθήκευση των παρελθουσών εμπειριών (κατάσταση, ενέργεια, ανταμοιβή, επόμενη κατάσταση), επιτρέποντας έτσι στον πράκτορα να μαθαίνει από ένα μεγαλύτερο σύνολο καταστάσεων παιχνιδιών και ενεργειών αντί μόνο των πιο πρόσφατων, αυτό βοηθά επίσης στην σταθεροποίηση της διαδικασίας μάθησης.

Οι τιμές Q ενημερώνονται χρησιμοποιώντας την μέθοδο καθοδικής κλίσης (gradient descent), όπου οι κλίσεις υπολογίζονται βάση της συνάρτησης απώλειας (calculate_q_value_loss), μετά η απώλεια που υπολογίζεται χρησιμοποιείται για την ενημέρωση των παραμέτρων του μοντέλου κατά την διάρκεια της εκπαίδευσης. Τέλος, η εξίσωση Bellman χρησιμοποιείται για την εκτίμηση των τιμών-στόχων Q, συμπεριλαμβάνοντας τις ανταμοιβές και τις εκπτωτικές τιμές Q της επόμενης κατάστασης.

Η στρατηγική εξερεύνησης που χρησιμοποιείται στην υλοποίηση είναι ο αλγόριθμος Epsilon-Greedy όπου αναλύσαμε σε προηγούμενο κεφάλαιο, συνοπτικά χρησιμοποιώντας αυτόν τον αλγόριθμο σημαίνει ότι ο πράκτορας είτε εκμεταλλεύεται την καλύτερη ενέργεια που γνωρίζει είτε εξερευνά μια νέα ενέργεια με βάση τον ρυθμό εξερεύνησης (το έψιλον) το όποιο μειώνεται (decays) όσο περνάει ο χρόνος, αυτό γίνεται για την μετάβαση από την εξερεύνηση του χώρου δράσης προς την εκμετάλλευση της μαθημένης πολιτικής.

Ο κύριος στόχος του πράκτορα στα πλαίσια του παιχνιδιού Σκορ 4 είναι να μάθει μια πολιτική που μεγιστοποιεί την αναμενόμενη συσσωρευτική ανταμοιβή (δηλαδή, να κερδίσει το παιχνίδι) επιλέγοντας την βέλτιστη ακολουθία ενεργειών (κινήσεων) από οποιαδήποτε δεδομένη κατάσταση παιχνιδιού.

4.2 Περιεχόμενο αρχείων κώδικα

Σε αυτή την υποενότητα, πραγματοποιείται μια λεκτική περιγραφή του τι περιέχει κάθε αρχείου κώδικα που χρησιμοποιείται στην υλοποίηση και τον ρόλο που έχουν σε κάθε κομμάτι της λειτουργίας της υλοποίησης.

4.2.1 Κύριο αρχείο κώδικα (main.py)

Στο αρχείο του κώδικα της main.py συντονίζεται η εκπαίδευση και η ανάπτυξη του πράκτορα. Εκμεταλλεύεται βιβλιοθήκες όπως την JAX για αποτελεσματικούς αριθμητικούς υπολογισμούς και ενσωματώνει τα υπόλοιπα αρχεία του προγράμματος: για την διαμόρφωση του περιβάλλοντος από το env.py, επαναλαμβανόμενη εμπειρία από το buffer.py και την αρχιτεκτονική του πράκτορα από το agent.py.

Στην αρχή, δημιουργεί ένα περιβάλλον παιχνιδιού με συγκεκριμένες ρυθμίσεις για την προεπεξεργασία καταστάσεων και τον έλεγχο νίκης. Στην συνέχεια, αρχικοποιεί έναν πράκτορα νευρωνικού δικτύου με μια καθορισμένη αρχιτεκτονική, περιλαμβανομένων συνελικτικών και πυκνών στρωμάτων και καθορίζει υπερπαραμέτρους όπως είναι η συνάρτηση απώλειας, ο ρυθμός μάθησης και η στρατηγική εξερεύνησης.

Δίνει την δυνατότητα ευελιξίας στην διαχείριση της διαδικασίας μάθησης του πράκτορα, επιτρέπει την φόρτωση ενός προηγούμενου εκπαιδευμένου μοντέλου για να παρακάμψει την φάση εκπαίδευσης ή την έναρξη μιας νέας διαδικασίας εκπαίδευσης. Ο βρόχος εκπαίδευσης είναι ουσιαστικά ο πράκτορας που παίζει παιχνίδια εναντίον του εαυτού του, κατά την διάρκεια των οποίων χρησιμοποιεί μια στρατηγική εξερεύνησης epsilon-greedy ισορροπώντας μεταξύ της εξερεύνησης νέων δράσεων και της εκμετάλλευσης γνωστών στρατηγικών. Οι εμπειρίες του πράκτορα αποθηκεύονται στο buffer αναπαραγωγής, από τον οποίο επιλέγονται μικρά δείγματα (minibatch) για την εκπαίδευση του νευρωνικού δικτύου. Αυτή η διαδικασία διακόπτεται περιοδικά για να πραγματοποιηθούν οι ενημερώσεις στο ποσοστό εξερεύνησης και για την εκτύπωση των μετρήσεων της προόδου εκπαίδευσης.

Μετά την ολοκλήρωση της εκπαίδευσης, το εκπαιδευμένο μοντέλο αποθηκεύεται. Παρέχει επίσης την λειτουργικότητα για να παίξει ένας ανθρώπινος παίκτης εναντίον του αποθηκευμένου μοντέλου του πράκτορα.

Οι σημαντικές συναρτήσεις που περιέχει η main.py:

- update_exploration_rate: προσαρμόζει τον ρυθμό εξερεύνησης με βάση την πρόοδο της εκπαίδευσης
- print_training_progress: εμφανίζει τις τρέχουσες μετρικές από την εκπαίδευση
- print_game_outcome: εμφανίζει το αποτέλεσμα ενός γύρου παιχνιδιού
- determine_ai_action: αποφασίζει την κίνηση του πράκτορα με βάση την τρέχουσα κατάσταση του παιχνιδιού
- play_against_ai: διαχειρίζεται τον βρόχο παιχνιδιού για το παίξιμο μεταξύ ανθρώπινου παίκτη και του πράκτορα
- prompt_user_for_move: συλλέγει μια κίνηση από τον ανθρώπινο παίκτη

4.2.2 Περιβάλλον πράκτορα (env.py)

Σε αυτό το αρχείο κώδικα υλοποιείται το περιβάλλον του Σκορ 4 στο όποιο θα αλληλοεπιδρά ο πράκτορας, το περιβάλλον διαθέτει προϋπολογισμένες γραμμές νίκης (χρησιμοποιώντας έναν τρισδιάστατο πίνακα της βιβλιοθήκης NumPy, const_lines) για πιο γρήγορο έλεγχο νίκης, προσαρμόσιμη προεπεξεργασία κατάστασης παιχνιδιού και μεθόδους για την κίνηση δίσκων, την αναίρεση κινήσεων και την επανεκκίνηση του παιχνιδιού.

Η κλάση ConnectFour περιλαμβάνει όλη την λογική του παιχνιδιού. Πραγματοποιεί την αρχικοποίηση του πίνακα του παιχνιδιού (με την κλήση της συνάρτησης __init__, η οποία ουσιαστικά δημιουργεί ένα νέο στιγμιότυπο - instance παιχνιδιού), την παρακολούθηση της

κορυφαίας διαθέσιμης σειράς σε κάθε στήλη και την διατήρηση της τρέχουσας κατάστασης του παιχνιδιού. Με την κλήση της συνάρτησης make_move ο παίκτης (σε αυτή την περίπτωση ο πράκτορας), πραγματοποιεί κίνηση και παράλληλα ελέγχει (αφού πραγματοποιηθεί η κίνηση) αν αποτελεί μια κίνηση που αποφέρει νίκη για τον παίκτη.

Το περιβάλλον διαθέτει και άλλες σημαντικές συναρτήσεις όπως την undo_move, η οποία επιτρέπει την αναίρεση της κίνησης, χρήσιμη για την εξερεύνηση διαφορετικών καταστάσεων. Η συνάρτηση reset_game καθαρίζει τον πίνακα για ένα νέο παιχνίδι και η συνάρτηση display_board αναπαριστά οπτικά την τρέχουσα κατάσταση του παιχνιδιού.

4.2.3 Αποθήκευση και διαχείριση εμπειριών (buffer.py)

Η δουλεία της κλάσης ExperienceReplayBuffer είναι να αποθηκεύει και να διαχειρίζεται τις εμπειρίες, οι οποίες αποτελούν τα ουσιώδη σημεία δεδομένων που προκύπτουν από τις αλληλεπιδράσεις του πράκτορα με το περιβάλλον του παιχνιδιού. Αυτές οι εμπειρίες αποτελούνται συνήθως από πλειάδες (tuples) που περιέχουν την κατάσταση (state) του παιχνιδιού, την ενέργεια (action) που πραγματοποίησε ο πράκτορας, την επόμενη κατάσταση (next state) που προέκυψε και την ανταμοιβή (reward) που λήφθηκε.

Η κλάση ExperienceReplayBuffer αποθηκεύει τις εμπειρίες σε ξεχωριστούς πίνακες NumPy για τις προηγούμενες καταστάσεις, ενέργειες, επόμενες καταστάσεις, ανταμοιβές, τα αποτελέσματα και τις ταυτότητες των παικτών (player IDs). Όταν παράγεται μια νέα εμπειρία, αποθηκεύεται στον δείκτη που καθορίζεται από τον μετρητή μνήμης (memory_counter). Αυτός ο μετρητής αυξάνεται με κάθε νέα εμπειρία, επιστρέφοντας στο 0 αν υπερβεί το μέγεθος του buffer, υποδηλώνοντας ότι ο buffer είναι γεμάτος και οι παλαιότερες εμπειρίες θα αντικατασταθούν. Η απόφαση για την αντικατάσταση βασίζεται στην αρχή του πρώτο-εισέρχεται-πρώτο-εξέρχεται (FIFO), η οποία είναι μια απλή αλλά αποτελεσματική λογική ως προς την διαχείριση της μνήμης, εξασφαλίζοντας ότι ο buffer περιέχει ένα μείγμα πρόσφατων αλλά και παλαιότερων εμπειριών.

Η επανάληψη εμπειρίας (experience replay) συμβάλλει σημαντικά στην σταθεροποίηση της διαδικασίας μάθησης. Με την αποθήκευση ενός χώρου επανάληψης παρελθουσών εμπειριών, το δίκτυο μπορεί να δειγματοληπτήσει ένα μίνι-σύνολο εμπειριών για εκπαίδευση, αντί να χρησιμοποιήσει μόνο την πιο πρόσφατη εμπειρία. Αυτή η προσέγγιση κόβει την συσχέτιση μεταξύ των διαδοχικών δειγμάτων μάθησης, χωρίς την προσέγγιση αυτή η ενημέρωση θα μπορούσε να οδηγηθεί σε υψηλή διακύμανση και ενδεχομένως να αποσταθεροποιήσει την διαδικασία μάθησης. Επιπλέον, η επανάληψη εμπειρίας επιτρέπει στο δίκτυο να μά-

θει από τις επιμέρους εμπειρίες πολλές φορές, βελτιώνοντας την αποδοτικότητα των δεδομένων και συμβάλλοντας στην σύγκλιση προς τις βέλτιστες πολιτικές.

Η υλοποίηση αυτή αντιμετωπίζει όλες τις εμπειρίες ισότιμα (δεν κάνει διακρίσεις), χωρίς να καθορίζει την σημαντικότητα της κάθε εμπειρίας. Ωστόσο σε μια ενδεχομένως επέκταση της υλοποίησης, θα μπορούσαν να εφαρμοστούν τεχνικές όπως η προτεραιοποιημένη επανάληψη εμπειρίας (Prioritized Experience Replay) η οποία θα έδινε προτεραιότητα στις εμπειρίες με βάση το σφάλμα της χρονικής διαφοράς (TD), διαθέτοντας περισσότερο χώρο στην μνήμη buffer και αυξάνοντας την συχνότητα δειγματοληψίας για εμπειρίες από τις οποίες ο πράκτορας θα μπορούσε να μάθει περισσότερα (δηλ. πιο ωφέλιμες εμπειρίες για την διαδικασία μάθησης).

Ο buffer υποστηρίζει την ικανότητα του δικτύου να μαθαίνει και να προσαρμόζεται στο περιβάλλον του παιχνιδιού με αρκετούς τρόπους:

- Επαύξηση δεδομένων: Με την (πιθανή) αναστροφή (flip) των καταστάσεων και των ενεργειών του πίνακα οριζόντια, ο διαχειριστής μνήμης εισάγει μια μορφή επαύξησης δεδομένων, η οποία βοηθάει το δίκτυο να γενικεύσει καλύτερα μαθαίνοντας ότι οι κατοπτρικές διαμορφώσεις του πίνακα είναι ουσιαστικά ίδιες όσον αφορά την στρατηγική.
- Διαφορετικά δείγματα εμπειρίας: Με την δειγματοληψία τυχαίων μίνι-παρτίδων (mini-batches) από το buffer εξασφαλίζεται ότι το δίκτυο εκτίθεται σε μια ευρεία γκάμα καταστάσεων, περιλαμβανομένων πολλών διαφόρων σταδίων του παιχνιδιού και διαφορετικών στρατηγικών αντιπάλων. Αυτή η ποικιλομορφία ενθαρρύνει το δίκτυο να μάθει σταθερές πολιτικές που είναι αποτελεσματικές σε διάφορα σενάρια.
- Αποσυσχέτιση διαδοχικών εμπειριών: Με την εκπαίδευση του δίκτυο σε μίνι-παρτίδες εμπειριών που δεν είναι διαδοχικά συσχετιζόμενες, το δίκτυο είναι λιγότερο πιθανό να υπερπροσαρμοστεί σε συγκεκριμένες ακολουθίες ενεργειών και καταστάσεων, προάγοντας μια πιο σταθερή και γενικευμένη διαδικασία μάθησης.

Ο διαχειριστής μνήμης έχει καίριο ρόλο στην διαδικασία μάθησης του δικτύου, διαχειρίζεται τις εμπειρίες με τρόπο που υποστηρίζει την αποτελεσματική μάθηση μέσω της επανάληψης εμπειρίας. Η σχεδίαση και η λειτουργικότητα του συμβάλλουν άμεσα στην ικανότητα του δικτύου να μαθαίνει βέλτιστες πολιτικές, σταθεροποιώντας την διαδικασία μάθησης και προάγοντας την ανάπτυξη γενικευμένων στρατηγικών.

4.2.4 Πλαίσιο για την υλοποίηση του πράκτορα (agent.py)

Ο κώδικας του αρχείου agent.py αποτελεί το πλαίσιο για την υλοποίηση του πράκτορα βαθιάς ενισχυτικής μάθησης, σχεδιασμένο ειδικά για εργασίες λήψης αποφάσεων. Ο πράκτορας χρησιμοποιεί το νευρωνικό δίκτυο για να προσεγγίσει την συνάρτηση Q, η οποία αντιστοιχίζει καταστάσεις (και ενέργειες) σε αναμενόμενες (expected) μελλοντικές ανταμοιβές. Το αρχείο δομείται γύρω από τα κύρια στοιχεία που καθορίζουν την αρχιτεκτονική του πράκτορα, την λειτουργικότητα και την διαδικασία μάθησης.

Η συνάρτηση calculate_q_value_loss υπολογίζει την απώλεια για την εκπαίδευση του δικτύου Q. Χρησιμοποιεί την εξίσωση Bellman για να υπολογίσει τις στοχευμένες τιμές Q (target Q-Values) και στην συνέχεια υπολογίζει το μέσο τετραγωνικό σφάλμα (mean squared error) μεταξύ των στοχευμένων τιμών Q και των τιμών Q που προβλεπόνται (predicted Q-Values) από το δίκτυο για τις ληφθείσες ενέργειες. Ως είσοδο παίρνει την predict_fn (η συνάρτηση πρόβλεψης του δίκτου Q), state_current και state_next (τρέχουσα και επόμενη κατάσταση), ενέργεια, ανταμοιβή και params (πρόσθετοι παράμετροι όπως ο παράγοντας έκπτωσης). Παράγει ως έξοδο την απώλεια, η οποία χρησιμοποιείται για την ενημέρωση των βαρών του δικτύου.

Η αφηρημένη διεπαφή BaseAgent αποτελεί μια κλάση που περιγράφει την δομή ενός πράκτορα βαθιάς ενισχυτικής μάθησης. Δηλώνει μεθόδους όπως την predict, update_model και select_action, οι οποίες πρέπει να υλοποιηθούν από τις υποκλάσεις.

Η κλάση NeuralNetworkAgent (η οποία κληρονομεί από την BaseAgent) υλοποιεί έναν πράκτορα βασισμένο σε νευρωνικό δίκτυο ικανό να προβλέπει ενέργειες, να ενημερώνει το μοντέλο του με βάση τις εμπειρίες και να επιλέγει ενέργειες με βάση την τρέχουσα κατάσταση. Μια από τις κύριες συναρτήσεις που διαθέτει είναι η initialize_model, η οποία ρυθμίζει το μοντέλο του νευρωνικού δικτύου χρησιμοποιώντας την μονάδα stax από την βιβλιοθήκη JAX, ορίζοντας την αρχιτεκτονική με βάση το σχήμα εισόδου (input shape), τα κρυφά επίπεδα (hidden layers) και το μέγεθος εξόδου (output size) επιπλέον προετοιμάζει την συνάρτηση απώλειας και τον βελτιστοποιητή (optimizer).

Η συνάρτηση predict επιστρέφει τις τιμές Q που προβλέφθηκαν από το δίκτυο για μια δεδομένη κατάσταση. Η συνάρτηση select_action επιλέγει μια ενέργεια με βάση την τρέχουσα κατάσταση, χρησιμοποιώντας την πολιτική epsilon-greedy για την εξερεύνηση.

Η συνάρτηση update_model ενημερώνει τις παραμέτρους του μοντέλου με βάση ένα μικρό δείγμα εμπειριών (minibatch) από την μνήμη, χρησιμοποιώντας την καθοδική κλίση (gradient descent) για την ελαχιστοποίηση της απώλειας. Η συνάρτηση simulate_future κάνει

εκτίμηση των μελλοντικών ανταμοιβών για μια δεδομένη κατάσταση και ενέργεια με την αναδρομική προσομοίωση μελλοντικών βημάτων.

Ο σκοπός της συνάρτησης compute_avg_loss είναι να υπολογίζει την μέση απώλεια του πράκτορα πάνω σε έναν συγκεκριμένο αριθμό δειγμάτων από την μνήμη. Αυτή η συνάρτηση είναι χρήσιμη για την αξιολόγηση της απόδοσης του πράκτορα κατά την διάρκεια που εκπαιδεύεται. Η είσοδος που παίρνει η συνάρτηση είναι ένα στιγμιότυπο (instance) της κλάσης NeuralNetworkAgent, την mem (η μνήμη του buffer που περιέχει τις εμπειρίες) και το miniBatchSize (το μέγεθος των δειγμάτων προς δειγματοληψία από την μνήμη).

Άρα ας συνοψίσουμε τον τρόπο που λειτουργεί η agent.py. Στην αρχή ξεκινάει με την 1) αρχικοποίηση: δημιουργείται ένα στιγμιότυπο του NeuralNetworkAgent με καθορισμένες παραμέτρους που ορίζουν την αρχιτεκτονική του νευρωνικού δικτύου και τις ρυθμίσεις εκπαίδευσης. Κατά την 2) διαδικασία μάθησης: εμπειρίες (μεταβάσεις καταστάσεων, ενέργειες, ανταμοιβές) συλλέγονται και αποθηκεύονται στον buffer κατά την διάρκεια του παιξίματος. Περιοδικά, ο πράκτορας δειγματοληπτεί μικρές παρτίδες εμπειριών από την μνήμη και τις χρησιμοποιεί για να ενημερώσει το μοντέλο του. Αυτό περιλαμβάνει τον υπολογισμό της απώλειας χρησιμοποιώντας την συνάρτηση calculate_q_value_loss και την εκτέλεση της διαβάθμισης κλίσης (gradient descent) για την προσαρμογή των βαρών του δικτύου. Ο πράκτορας χρησιμοποιεί την μέθοδο select_action για να επιλέξει ενέργειες είτε τυχαία (για εξερεύνηση) είτε βασιζόμενος στις προβλέψεις του νευρωνικού του δικτύου (εκμετάλλευση), προσαρμόζοντας την πολιτική του με την πάροδο του χρόνου καθώς μαθαίνει. Και τέλος για το μέρος της 3) αξιολόγησης: η απόδοση του πράκτορα παρακολουθείται υπολογίζοντας την μέση απώλεια πάνω σε ένα σύνολο εμπειριών, βοηθώντας στην εκτίμηση του πόσο καλά ο πράκτορας μαθαίνει να προβλέπει μελλοντικές ανταμοιβές.

5 Εγχειρίδιο υλοποίησης

Αυτό το κεφάλαιο ουσιαστικά αποτελεί το εγχειρίδιο που παρέχει τις οδηγίες για την εγκατάσταση, την παραμετροποίηση και την εκτέλεση της υλοποίησης. Δίνεται επίσης και ένα παράδειγμα παιξίματος εναντίον του πράκτορα. Ο κώδικας της υλοποίησης είναι ανεβασμένος στο προσωπικό GitHub.

5.1 Απαιτήσεις

Για την εύρυθμη λειτουργία της υλοποίησης, προτείνεται να χρησιμοποιήσετε το λειτουργικό σύστημα των Windows 10, την εγκατάσταση της εκδόσεως 3.7 της Python και κάποιο περιβάλλον ανάπτυξης με παρόμοιες λειτουργίες όπως του PyCharm Professional. Επίσης δεν απαιτείται να υπάρχει διακριτή μονάδα επεξεργασίας γραφικών (dGPU) παρά μόνο να υπάρχει μια ενσωματωμένη μονάδα επεξεργασίας γραφικών (iGPU) του επεξεργαστή.

Πριν τρέξουμε (ή παραμετροποιήσουμε) την υλοποίηση, βεβαιωνόμαστε ότι έχουμε εγκαταστήσει την Python 3.7 στο σύστημα. Μετέπειτα ακολουθούμε τα παρακάτω βήματα για να ρυθμίσουμε το περιβάλλον και να εγκαταστήσουμε τις απαιτούμενες βιβλιοθήκες:

1. Ανοίγουμε το PyCharm Professional και δημιουργούμε ένα νέο έργο.
2. Ανοίγουμε το τερματικό που διαθέτει το PyCharm πηγαίνοντας στο View > Tool Windows > Terminal (ή πατώντας ALT + F12).
3. Πραγματοποιούμε την εγκατάσταση των απαιτούμενων βιβλιοθηκών εκτελώντας την ακόλουθη εντολή στο τερματικό: `pip install Keras-Applications==1.0.8 Keras-Preprocessing==1.1.2 Markdown==3.4.4 MarkupSafe==2.1.5 Pillow==9.5.0 Werkzeug==2.2.3 absl-py==2.1.0 astor==0.8.1 build==1.1.1 colorama==0.4.6 cycler==0.11.0 etils==0.9.0 flatbuffers==2.0.7 fonttools==4.38.0 gast==0.2.2 google-pasta==0.2.0 grpcio==1.62.0 h5py==2.10.0 importlib-metadata==6.7.0 importlib-resources==5.12.0 jax==0.3.14 jaxlib==0.3.14 kiwisolver==1.4.5 matplotlib==3.5.3 numpy==1.21.6 opt-einsum==3.3.0 packaging==23.2 pip==23.2.1 protobuf==3.20.3 pyparsing==3.1.1 pyproject-hooks==1.0.0 python-dateutil==2.9.0.post0 scipy==1.7.3 setuptools==68.0.0 six==1.16.0 tensorboard==1.15.0 tensorflow-directml==1.15.8 tensor-`

```
flow-estimator==1.15.1    termcolor==2.3.0    tomli==2.0.1    typing-
extensions==4.7.1 wheel==0.41.2 wrapt==1.16.0 zipp==3.15.0
```

5.2 Παραμετροποίηση

Προτού εκτελέσουμε το μοντέλο, θα χρειαστεί η ρύθμιση (εφόσον επιθυμούμε να πειραματιστούμε με το μοντέλο) των παραμέτρων, στον παρακάτω πίνακα επεξηγείται η κάθε μια παράμετρος.

Πίνακας 1: Παράμετροι υλοποίησης και επεξήγηση αυτών.

Παράμετρος	**Επεξήγηση**
load_existing_model	Εάν θα φορτώσουμε ένα υπάρχον μοντέλο.
save_trained_model	Εάν θα αποθηκεύσουμε το μοντέλο μετά το πέρας της εκπαίδευσης.
number_of_training_steps	Ο αριθμός των βημάτων για την εκπαίδευση του μοντέλου.
memory_size_of_buffer	Το μέγεθος της μνήμης.
display_games	Εάν θα εμφανίζονται κάποια από τα παιχνίδια κατά την διάρκεια της εκπαίδευσης.
epsilon_start	Αποτελεί την αρχική τιμή για το έψιλον στην στρατηγική epsilon-greedy.
epsilon_end	Είναι η τελική τιμή που μπορεί να πάρει το έψιλον.
epsilon_decay_steps	Ο αριθμός βημάτων για την εξασθένηση του έψιλον.
learning_rate	Ρυθμός μάθησης για το μοντέλο.
model_name	Όνομα για την αποθήκευση του εκπαιδευμένου μοντέλου.

Η παραμετροποίηση του μοντέλου αποτελεί μια πειραματική και χρονοβόρα διαδικασία αλλά παράλληλα είναι και το κλειδί για την κατανόηση και την βελτίωση της απόδοσης του μοντέλου. Οι αρχικές παράμετροι που έχουν στηθεί προσφέρουν μια πολύ ικανοποιητική ικανότητα παιξίματος.

5.3 Εκτέλεση

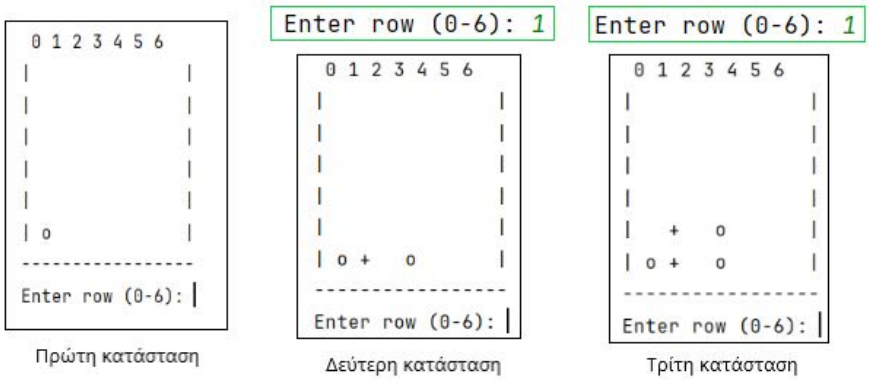
Ακολουθεί μια επεξήγηση ως προς την εκτέλεση της υλοποίησης:

1. Πρώτα ορίζουμε/αλλάζουμε τις παραμέτρους που βρίσκονται στην αρχή του αρχείου της main.py.
2. Εκτελούμε την υλοποίηση κάνοντας κλικ στο κουμπί ‘Εκτέλεση’ στην PyCharm (ή πατώντας Shift + F10).

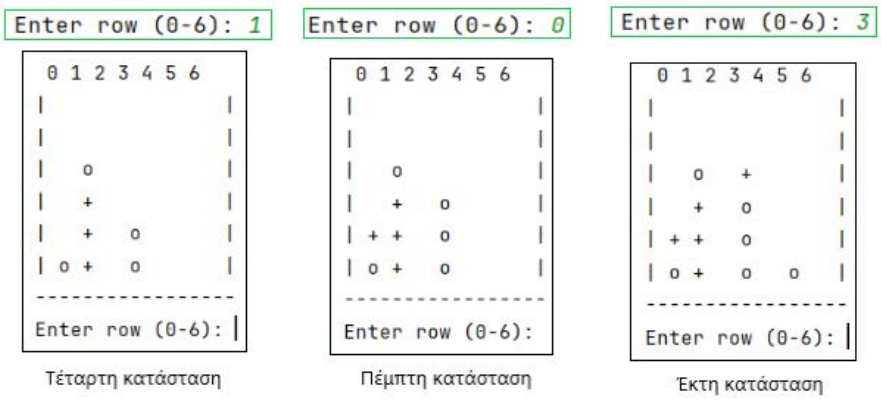
Αμέσως μετά ξεκινάει η εκπαίδευση και το μοντέλο θα αρχίσει να μαθαίνει πώς να παίζει Σκορ 4. Ανάλογα με τον αριθμό των βημάτων εκπαίδευσης που έχουμε ορίσει, αυτή η διαδικασία μπορεί να πάρει κάποιο χρόνο. Ευνοείτο είναι ότι όσο περισσότερο εκπαιδευτεί το μοντέλο τόσο καλύτερο θα είναι στο να παίζει Σκορ 4 (δηλ. να έχει μάθει μια καλή πολιτική). Εάν η παράμετρος display_games είναι ορισμένη σε True, θα μπορούμε να παρακολουθούμε τα παιχνίδια που παίζονται από το μοντέλο κατά την διάρκεια της εκπαίδευσης. Μόλις ολοκληρωθεί η εκπαίδευση, μπορούμε να παίξουμε ενάντια του μοντέλου.

5.3.1 Παράδειγμα παιξίματος

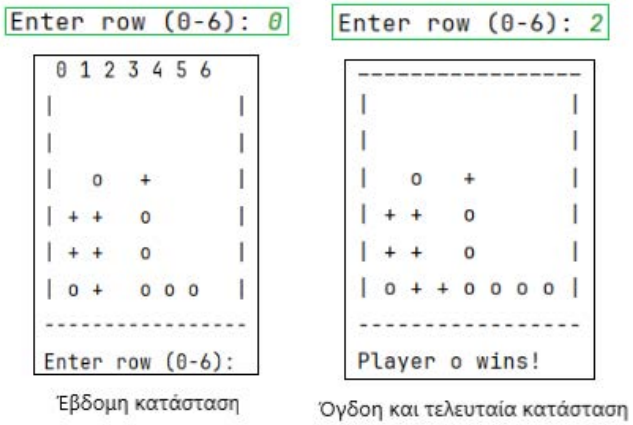
Όταν παίζουμε εναντίον του πράκτορα, πρέπει να παρατηρούμε την στρατηγική που ακολουθεί για να βελτιώσουμε το παιχνίδι. Πειραματιζόμαστε με διαφορετικές αρχικές κινήσεις για να δούμε πως προσαρμόζεται ενάντια σε αυτές. Ακολουθεί ένα παράδειγμα παιξίματος εναντίον του. Ο παίκτης ο είναι ο εκπαιδευμένος πράκτορας ενώ ο + είναι ο χρήστης, ξεκινάει με την πρώτη κίνηση ο πράκτορας.



Εικόνα 7: Πρώτο σετ εικόνων παιξίματος εναντίον του πράκτορα.



Εικόνα 8: Δεύτερο σετ εικόνων παιξίματος εναντίον του πράκτορα.



Εικόνα 9: Τρίτο σετ εικόνων παιξίματος εναντίον του πράκτορα.

Οπότε κάθε φορά που παίζει ο πράκτορας απεικονίζεται η κατάσταση του παιχνιδιού στην κονσόλα και ζητάει είσοδο από τον παίκτη (δηλ. την στήλη που θέλει να ρίξει το σύμβολο του – από 0 έως 6). Ο πράκτορας κατάφερε να μας νικήσει σε αυτή την παρτίδα παιχνιδιού, παίζει ρόλο και η υπομονή που έχει ο παίκτης ώστε να αναζητήσει τις καλύτερες κινήσεις που θα μπορούσε να κάνει.

5.4 Αντιμετώπιση προβλημάτων

Καταγραφή κάποιον πιθανών προβλημάτων που μπορεί να προκύψουν όταν πάμε να τρέξουμε την υλοποίηση:

- Σφάλματα κατά την εκκίνηση: πρέπει να βεβαιωθούμε ότι όλες οι βιβλιοθήκες είναι εγκατεστημένες με τις καθορισμένες εκδόσεις για να αποφευχθούν θέματα συμβατότητας.
- Απόδοση: η απόδοση της εκπαίδευσης εξαρτάται από την CPU. Μια πιο ισχυρή CPU θα οδηγήσει σε ταχύτερους χρόνους εκπαίδευσης.

- Το μοντέλο δεν βελτιώνεται: εάν το μοντέλο δεν βελτιώνεται, μπορείτε να αλλάξετε τον ρυθμό μάθησης, τις τιμές έψιλον ή να αυξήσετε τον αριθμό των βημάτων εκπαίδευσης. Πολύ υψηλές ή πολύ χαμηλές τιμές μπορεί να εμποδίσουν την μάθηση.

Σημειώσεις σχετικά με την απόδοση: η εκπαίδευση με την CPU μπορεί να είναι πολύ βαριά υπολογιστικά. Μια διακριτή μονάδα επεξεργασίας μπορεί να επιταχύνει σημαντικά τους χρόνους εκπαίδευσης αλλά δεν απαιτείται για την εκτέλεση αυτής της υλοποίησης (προτείνεται όμως). Η διάρκεια της εκπαίδευσης θα διαφέρει ανάλογα με τον αριθμό των βημάτων εκπαίδευσης και την πολυπλοκότητα του μοντέλου. Η υπομονή αποτελεί «κλειδί» στην διαδικασία.

6 Συμπεράσματα

Οι διαδικασίες μάθησης των ανθρώπων και των πρακτόρων βαθιάς ενισχυτικής μάθησης (DRL Agents), παρόλο που διαφέρουν ως προς στους μηχανισμούς και τις βασικές αρχές τους, μοιράζονται την ίδια προσέγγιση, αυτή της μάθησης μέσω μιας διαδικασίας δοκιμής και σφάλματος (trial and error). Αυτή η διαδικασία περιλαμβάνει την λήψη αποφάσεων, την εμπειρία που παίρνουμε μέσω αποτελεσμάτων και την προσαρμογή των συμπεριφορών με βάση αυτών για την βελτίωση της μελλοντικής απόδοσης.

Μερικές από τις προκλήσεις κατά την δημιουργία μιας τέτοιας υλοποίησης περιλαμβάνει την επιλογή ενός αλγορίθμου που συμφωνεί με την πολυπλοκότητα του παιχνιδιού και την φύση του χώρου δράσης (διακριτός ή συνεχής), η επιλογή του παιχνιδιού επίσης είναι σημαντική, καθώς πρέπει να παρέχει αρκετή πολυπλοκότητα για να μπορεί να δείξει την δύναμη της βαθιάς ενισχυτικής μάθησης, η αρχιτεκτονική του νευρωνικού δικτύου παίζει επίσης πολύ σημαντικό ρόλο στην απόδοση του τελικού μοντέλου, η επιλογή σωστών βιβλιοθηκών για την υποστήριξη της δημιουργίας ενός τέτοιου μοντέλου, το σύστημα ανταμοιβής είναι σημαντικό για την καθοδήγηση της διαδικασίας μάθησης και τέλος ο τρόπος αναπαράστασης της κατάστασης του παιχνιδιού μπορεί να επηρεάσει σημαντικά την αποδοτικότητα και τα αποτελέσματα της μάθησης.

Μια επιτυχημένη εφαρμογή της βαθιάς ενισχυτικής μάθησης για κάποιο παιχνίδι απαιτεί την αντιμετώπιση αυτών των προκλήσεων μέσω προσεκτικού σχεδιασμού και πειραματισμού (θα μπορούσε να πει κάνεις και μέσω μιας διαδικασίας trial and error). Κατανοώντας τις λεπτομέρειες κάθε πρόκλησης, από την επιλογή αλγορίθμου μέχρι την αναπαράσταση των δεδομένων, επιτυγχάνεται η δημιουργία αξιόπιστων μοντέλων DRL που επιδεικνύουν περίπλοκες, έξυπνες συμπεριφορές σε σύνθετα περιβάλλοντα παιχνιδιών.

Ενώ η υλοποίηση είναι αποτελεσματική, κάποιες προσθήκες θα μπορούσαν να αυξήσουν σημαντικά την χρηστικότητα της. Ένα από αυτά θα ήταν η ενσωμάτωση μηχανισμών οπτικής ανατροφοδότησης, όπως γραφήματα (π.χ. στο τέλος κάθε συνεδρίας εκπαίδευσης). Αυτές οι οπτικοποιήσεις θα μπορούσαν να παρέχουν άμεσες, κατανοητές διορατικότητες σχετικά με την απόδοση και την πρόοδο του μοντέλου. Επιπλέον, η ανάπτυξη μιας γραφικής διεπαφής χρήστη θα επέτρεπε στους χρήστες να αλληλοεπιδρούν πιο εύκολα με το μοντέλο. Με αυτό θα εξαλειφόταν η ανάγκη για άμεσο πείραγμα του κώδικα (π.χ. για να αλλάξουμε τις

παραμέτρους), διευρύνοντας την χρήση του εργαλείου σε ένα ευρύτερο κοινό χωρίς να χρειάζεται να έχουν εμπειρία στον προγραμματισμό, ενισχύοντας έτσι την εκπαιδευτική χρησιμότητα της υλοποίησης.

Βιβλιογραφία

[1] Morales, M., & Isbell, C. (2020). Grokking deep reinforcement learning. Manning Publications.

[2] Sutton, R. S., & Barto, A. (2018). Reinforcement learning : an introduction. The Mit Press.

[3] Sewak, M. (2019). Deep Reinforcement Learning.