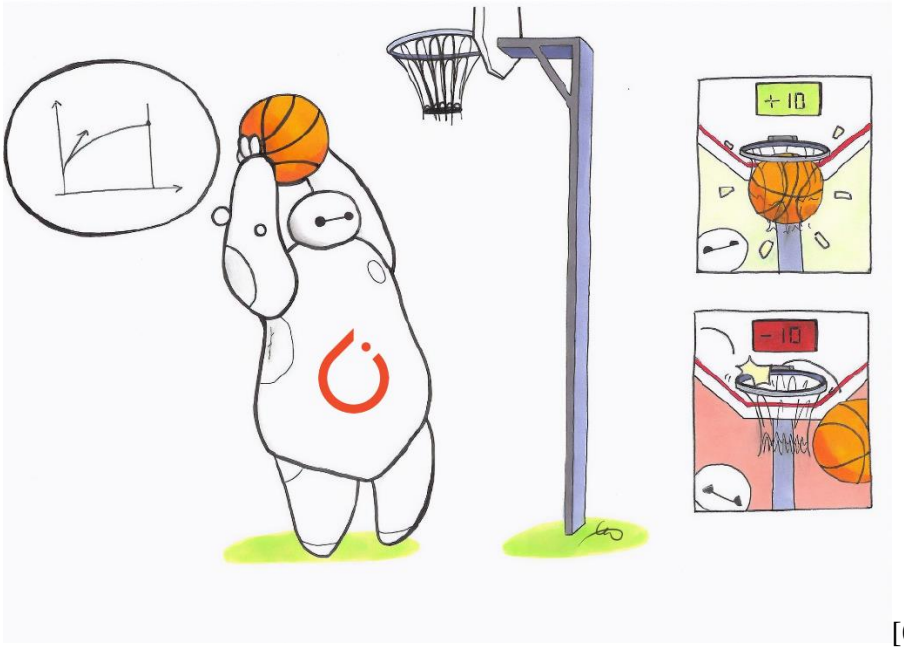
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Εκπαίδευση πρακτόρων ενισχυτικής μάθησης σε περιβάλλον παιχνιδιού»



[67]

Της φοιτήτριας
Σιάνδρης Ελπίδας
Αρ. Μητρώου: 174898

Επιβλέπων
Ονοματεπώνυμο Βάσσιος Βασίλειος
Βαθμίδα Ακαδημαϊκός Υπότροφος

Ημερομηνία: 31/8/2022

Τίτλος Δ.Ε. Εκπαίδευση πρακτόρων ενισχυτικής μάθησης σε περιβάλλον παιχνιδιού
Κωδικός Δ.Ε. 8537
Ονοματεπώνυμο φοιτήτριας: Σιάνδρη Ελπίδα
Ονοματεπώνυμο εισηγητή: Βάσσιος Βασίλειος
Ημερομηνία ανάληψης Δ.Ε. 13.03.2022
Ημερομηνία περάτωσης Δ.Ε. …

Βεβαιώνω ότι είμαι η συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία της φοιτήτριας Σιάνδρης Ελπίδας που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης, ο συγγραφέας/δημιουργός εκχωρεί στο Διεθνές Πανεπιστήμιο της Ελλάδος άδεια χρήσης του δικαιώματος αναπαραγωγής, δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσης της εργασίας διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος. Η ανοικτή πρόσβαση στο πλήρες κείμενο της εργασίας, δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού, ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, πώληση, εμπορική χρήση, διανομή, έκδοση, μεταφόρτωση (downloading), ανάρτηση (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Στην οικογένεια μου»

Πρόλογος

Αρκετοί είναι οι λόγοι που επέλεξα να ασχοληθώ με την εκπαίδευση των πρακτόρων ενισχυτικής μάθησης σε περιβάλλον παιχνιδιού. Αρχικά, ένας λόγος αποτελεί το γεγονός πως ποτέ δεν μου είχε δοθεί η ευκαιρία να ασχοληθώ με τον τομέα της Μηχανικής Μάθησης. Επιλέγοντας, αυτό το θέμα, κατάφερα να διευρύνω τους ορίζοντες μου και να αποκτήσω μια εμπειρία πάνω στον τομέα της ενισχυτικής μάθησης. Έτσι, πλέον μπορώ να αποφασίσω, εάν με ενδιαφέρει και επαγγελματικά να ασχοληθώ με τον τομέα της Μηχανικής Μάθησης.

Περίληψη

Με το πέρασμα των χρόνων, υπήρξε μια ραγδαία ανάπτυξη στον τομέα της Μηχανικής Μάθησης. Η χρήση της πραγματοποιήθηκε σε πολλούς τομείς, όπως την ιατρική, την όραση υπολογιστών και σε παιχνίδια. Ο χώρος της μηχανικής μάθησης και ειδικότερα της ενισχυτικής μάθησης, βασιζόμενος στο παραπάνω δόγμα, δημιουργεί πράκτορες οι οποίοι δύναται να μαθαίνουν διαμέσου της συνεχούς αλληλεπίδρασης με το περιβάλλον. Βάση αυτού, η παρούσα διπλωματική εργασία πραγματεύεται την ανάπτυξη αλγορίθμου βαθιάς ενισχυτικής μάθησης σε περιβάλλον παιχνιδιού. Στα πλαίσια της διπλωματικής μελετιέται ο όρος Ενισχυτικής μάθησης και ο όρος βαθιά ενισχυτική μάθηση μαζί με τους αλγορίθμους του, αντίστοιχα. Σε αυτή τη διπλωματική, θα εκτελεστούν δύο παιχνίδια ενισχυτικής μάθησης με σκοπό τη δημιουργία ενός απλού περιβάλλοντος ενισχυτικής μάθησης και την εκπαίδευση ενός πράκτορα βαθιάς Ενισχυτικής Μάθησης στα πλαίσια του κάθε παιχνιδιού. Όσον αφορά, την υλοποίηση της διπλωματικής, πρόκειται για την υλοποίηση δύο παιχνιδιών ενισχυτικής μάθησης, του σκορ 4 και του Sudoku, όπου σε κάθε υλοποίηση παιχνιδιού υπάρχει ένα συμπέρασμα, το οποίο εκτιμά το πόσο καλή ήταν η εκπαίδευση του πράκτορα στο παιχνίδι και πως ήταν η απόδοση του. Τέλος, εξετάζεται και η περίπτωση η απόδοση του πράκτορα που χρησιμοποιείται στην προπόνηση μερικές φορές να μειώνει το αποτέλεσμα .

Λέξεις-κλειδιά Μηχανική μάθηση · Βαθιά ενισχυτική μάθηση · Πράκτορας · Εκπαίδευση · Απόδοση

«Training reinforcement learning agents in a game environment»

«Siandri Elpida»

Abstract

Over the years, there has been a rapid growth in the field of Machine Learning. It has been used in many fields such as medicine, computer vision and gaming. The field of machine learning and in particular reinforcement learning, based on the above doctrine, creates agents that can learn through continuous interaction with the environment. Based on this, this thesis deals with the development of a deep reinforcement learning algorithm in a game environment. In the context of the diploma, the term Reinforcement learning and the term deep reinforcement learning together with its algorithms are studied, respectively. In this thesis, two reinforcement learning games will be run to create a simple reinforcement learning environment and train a deep reinforcement learning agent in the context of each game. Regarding the implementation of diplomacy, it is about the implementation of two reinforcement learning games, score 4 and Sudoku, where in each game implementation there is a conclusion, which estimates how well the agent was trained in the game and how was his performance. Finally, the case that the performance of the agent used in training sometimes reduces the result is considered.

Keywords: Machine learning · Deep reinforcement learning · agent · training · performance

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Βάσσιο για την εμπιστοσύνη που μου έδειξε σχετικά με την εκπόνηση της παρούσας εργασίας και την ευκαιρία να εμβαθύνω τις γνώσεις μου στον τομέα της Μηχανικής Μάθησης. Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά και τους φίλους μου για την υποστήριξη τους.

Περιεχόμενα

Κατάλογος Εικόνων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Στις μέρες μας, τα παιχνίδια αποτελούν το ιδανικό υλικό για την υλοποίηση έρευνας και δοκιμών με σύγχρονες μεθόδους τεχνητής νοημοσύνης και μηχανικής μάθησης. Μία από τις διαδικασίες ενισχυτικής μάθησης αποτελεί η αυτοεκπαίδευση. Με αυτή την τεχνική ο πράκτορας είναι σε θέση να παίξει το παιχνίδι είτε με τον εαυτό του είτε με άλλον πράκτορα, όσες φορές θελήσει. Αναλόγως τις ενέργειες που εκτελεί κάθε φορά στο παιχνίδι, θα έχει και τις αντίστοιχες ανταμοιβές για να βελτιώσει σταδιακά τις επιλογές κινήσεών του στο παιχνίδι. [2]

Στην εργασία αυτή εκτελείται η εκπαίδευση ενός πράκτορα που θα παίζει sudoku και connect4, όπου η αλληλεπίδραση του παίκτη με το περιβάλλον θα γίνεται με τη χρήση μιας τυπικής Μαρκοβιανής Διαδικασίας. Ειδικότερα, ο πράκτορας μας μαθαίνει από την έκβαση των παρτίδων αυτοεκπαίδευσης, χρησιμοποιώντας ένα τεχνητό νευρωνικό δίκτυο για την εκμάθηση της συνάρτησης αξιολόγησης των παιχνιδιών αυτών.

Στα επόμενα κεφάλαια γίνεται μια ειδική ανασκόπηση. Συγκεκριμένα, αναφέρεται η γενική επισκόπηση της ενισχυτικής μάθησης, της βαθιάς ενισχυτικής μάθησης και οι αλγόριθμοι τους αντίστοιχα. Επίσης, αναλύονται εις βάθους οι αλγόριθμοι που χρησιμοποιήθηκαν, παρουσιάζεται η εκτέλεση της εκπαίδευσης του πράκτορα και φυσικά αξιολογείται η εκπαίδευση των πρακτόρων σε κάθε παιχνίδι. Στο τέλος της εργασίας, συνοψίζονται τα συμπεράσματα που προέκυψαν καθ’ όλη τη διάρκεια της εκπόνησης της εργασίας και προτείνονται πιθανές βελτιώσεις των πρακτόρων και μελλοντικές επεκτάσεις της ερευνητικής διαδικασίας.

Κεφάλαιο 2ο: Η Βιομηχανική Επανάσταση, η Γνώση και οι Υπολογιστές

Η Βιομηχανική Επανάσταση και η Ανάπτυξη των Υπολογιστών άρχισε να αντικαθιστά γρήγορα την ανθρώπινη εργασία [8]. Αυτό είχε ως αποτέλεσμα να δημιουργηθούν νέες βιομηχανίες και κοινωνίες, αλλάζοντας δραστικά κάθε πτυχή της ανθρώπινης ζωής. Συγκεκριμένα, η ζωή την ανθρώπων βελτιώθηκε αλλά η τεχνολογία των υπολογιστών θα έπρεπε να υλοποιεί όλες τις εργασίες που απαιτούν ανθρώπινη σκέψη. Ωστόσο, το μόνο που μπορεί να εκπληρώσει με ακρίβεια είναι ο υπολογισμός, διότι ένας υπολογιστής δεν διαθέτει εγκέφαλο, δηλαδή θορυβώδεις οργανικούς νευρώνες με απρόβλεπτα μοτίβα πυροδότησης, και κριτική σκέψη αλλά λειτουργεί χρησιμοποιώντας ντετερμινιστικές δυαδικές πύλες που υλοποιούνται σε πυρίτιο. Ακόμη, οι υπολογιστές στηρίζονται και εκτελούνται από προγράμματα για την εκτέλεση υπολογισμών βασισμένα σε bits. Αντίθετα, όλα τα νευρωνικά προγράμματα που εκτελούνται στον εγκέφαλο μαθαίνονται μέσω επαναλαμβανόμενης αλληλεπίδρασης με το περιβάλλον, αντί να παρέχονται από έναν προγραμματιστή.[7] Για αυτό τον λόγο, ξεκίνησε και συνεχίζεται να αναπτύσσεται η τεχνητή νοημοσύνη, η οποία έχει επικεντρωθεί στην αναδημιουργία συλλογιστικών ικανοτήτων που μοιάζουν με τις ικανότητες ενός ανθρώπου.[9]

Μια εναλλακτική μέθοδος για την παροχή γνωστικών ικανοτήτων στα ρομπότ είναι η μηχανική μάθηση (ML). Σημαντικό αποτελεί το γεγονός, πως η μηχανική μάθηση στηρίζεται στην περιγραφή ενός συνόλου από κανόνες εκμάθησης, που μπορεί να χρησιμοποιήσει ο αλγόριθμος για να αντλήσει κανόνες απόφασης από ένα επισημασμένο σύνολο δεδομένων παραδειγμάτων.[10] Για παράδειγμα, με την εξέλιξη της μηχανικής μάθησης, πλέον υπάρχουν συστήματα που μπορούν να καθορίσουν ποιο ζώο υπάρχει σε μια εικόνα, αφού για να καταλήξουν σε αυτό το συμπέρασμα, έχουν δοθεί αρκετά παραδείγματα για να μπορούν να ανταποκριθεί σωστά.

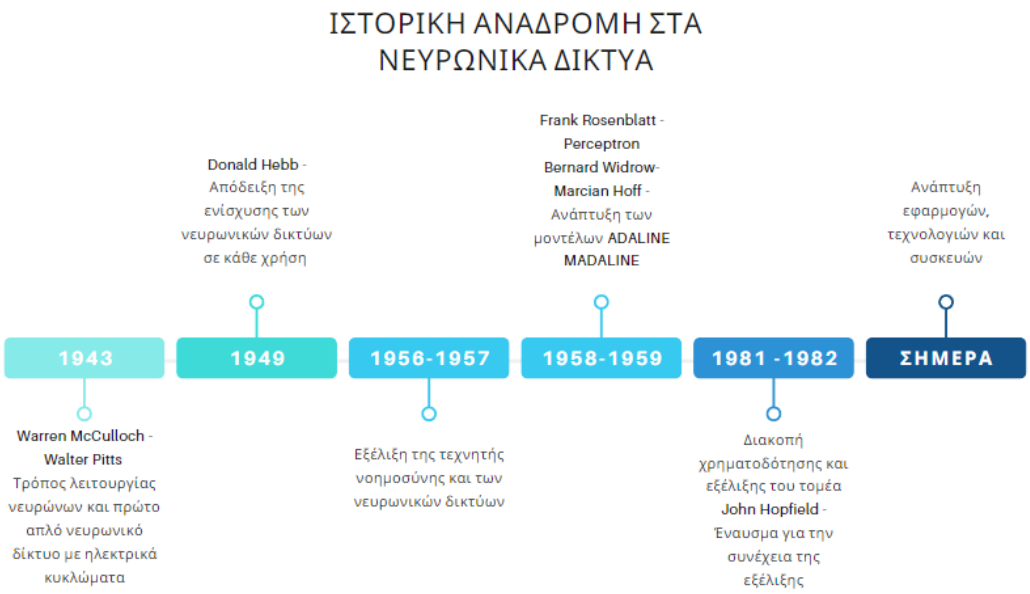
Κεφάλαιο 3ο: Νευρωνικά Δίκτυα

Σύμφωνα με τα παραπάνω, η δημιουργία ενός νευρωνικού δικτύου ξεκίνησε, όταν έγινε κατανοητό πως ο ανθρώπινος εγκέφαλος σκέφτεται και υπολογίζει εντελώς διαφορετικά από τον παραδοσιακό ψηφιακό υπολογιστή. Αυτό, οδήγησε στην ανάπτυξη και στην εξέλιξη των τεχνητών νευρωνικών δικτύων, όπου ο εγκέφαλος ενός νευρωνικού δικτύου είναι ένας πολύ εξελιγμένος, παράλληλος, μη γραμμικός υπολογιστής (σύστημα επεξεργασίας πληροφοριών) και έχει την ικανότητα να οργανώνει τους νευρώνες που αποτελούν τα δομικά του στοιχεία έτσι ώστε να μπορεί να πραγματοποιεί ορισμένους υπολογισμούς, Για παράδειγμα, έχει την ικανότητα αντίληψης, τον έλεγχο ενός κινητήρα πολύ πιο γρήγορα από έναν ψηφιακό υπολογιστή. Στόχος ενός νευρωνικού δικτύου είναι μετά από κάθε επεξεργασία του, να κατατοπίσει και να προσδώσει οποιαδήποτε πληροφορία λάβει στον χρήστη με σαφήνεια και με ταχύτητα. [7]

3.1 Ιστορική Αναδρομή στα Νευρωνικά Δίκτυα

Η εξέλιξη ενός τεχνητού νευρωνικού δικτύου ξεκίνησε πρώτα από τη μελέτη του ανθρώπινου εγκεφάλου. Ειδικότερα, το 1943 ο Warren McCulloch, ένας νευροφυσιολόγος, και ο μαθηματικός, Walter Pitts, ξεκίνησαν για πρώτη φορά μια μελέτη σχετικά με τον τρόπο λειτουργίας των νευρώνων και μάλιστα διαμόρφωσαν το πρώτο απλό νευρωνικό δίκτυο με ηλεκτρικά κυκλώματα. Το 1949, ο Donald Hebb ενίσχυσε την έννοια των νευρώνων στο βιβλίο του, The Organization of Behavior, στο οποίο επισήμανε ότι τα νευρικά μονοπάτια ενισχύονται κάθε φορά που χρησιμοποιούνται. Αργότερα το 1950, ο Nathanial Rochester προσομοίωσε για πρώτη φορά ένα νευρωνικό δίκτυο. Το 1956 το καλοκαιρινό ερευνητικό πρόγραμμα Dartmouth έδωσε ώθηση τόσο στην τεχνητή νοημοσύνη όσο και στα νευρωνικά δίκτυα και αυτό τόνωσε την έρευνα στην τεχνητή νοημοσύνη και στο επίπεδο της νευρωνικής επεξεργασίας του εγκεφάλου. Έτσι, το 1957, ο John von Neumann πρότεινε τη μίμηση απλών λειτουργιών νευρώνων χρησιμοποιώντας τηλεγραφικούς ηλεκτρονόμους ή σωλήνες κενού. Το 1958, ο Frank Rosenblatt, νευροβιολόγος του Cornell, κατάφερε να εξελίξει το Perceptron, ένα είδος τεχνητού νευρωνικού δικτύου, επειδή ερεύνησε τη λειτουργία του ματιού της μύγας. Συγκεκριμένα, η δομή του ματιού μιας μύγας περιέχει πολλούς νευρώνες, στους οποίους γίνεται το μεγαλύτερο μέρος της επεξεργασίας των αντανακλαστικών της.[6] Με την έρευνα του Frank Rosenblatt, αποδείχθηκε πως ο αλγόριθμος Perceptron, βασισμένος στον αρχικό νευρώνα MCP, είναι ένας αλγόριθμος που επιτρέπει στους νευρώνες να μαθαίνουν και να επεξεργάζονται στοιχεία στο σετ εκπαίδευσης ένα κάθε φορά. [14]Πλέον, μέχρι και σήμερα το Perceptron που αναπτύχθηκε ως αποτέλεσμα αυτής της μελέτης, είναι το παλαιότερο νευρωνικό δίκτυο που χρησιμοποιείται ακόμα σήμερα. Ένα χρόνο αργότερα, το 1959, ο Bernard Widrow και ο Marcian Hoff του Stanford ανέπτυξαν μοντέλα που ονόμασαν ADALINE και MADALINE. Το MADALINE είναι ένα πρώιμο τεχνητό νευρωνικό δίκτυο και παρέχει ένα προσαρμοστικό φίλτρο που απαλλάσσει τις ηχώ της τηλεφωνικής γραμμής, το οποίο ακόμη βρίσκεται σε εμπορική χρήση. Σύμφωνα με αυτή την έρευνα, αποδείχθηκε ότι η διαφορά μεταξύ της Adaline και του perceptron είναι ότι το perceptron είναι περιορισμένο στη χρήση του. Δυστυχώς. Μέχρι και το 1981 δεν υπήρχε κάποια εξέλιξη σε αυτόν τον τομέα επειδή ο φόβος και οι ανεκπλήρωτες επιθυμίες του λαού διέκοψαν την χρηματοδότηση για να συνεχιστούν οι έρευνες στα νευρωνικά δίκτυα. Όμως το 1982, ο John Hopfield παρουσίασε τη δική του προσέγγιση της εφαρμογής των νευρωνικών δικτύων πάνω σε συσκευές, η οποία έδωσε το έναυσμα στην επιστημονική κοινότητα να επιτρέψει τους επιστήμονες την συνέχεια στην μελέτη των νευρώνων. Αξίζει να σημειωθεί πως το 1987, το Διεθνές Συνέδριο του Ινστιτούτου Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών (IEEE)προσέλκυσε αρκετούς συμμετέχοντες για τα νευρωνικά δίκτυα Από τότε οι επιστήμονες αποδέχτηκαν πως η εξέλιξη των

νευρωνικών δικτύων θα τους βοηθήσει στην μελλοντική ανάπτυξη εφαρμογών, τεχνολογιών και συσκευών.[6]



Εικόνα 1: Ιστορική Αναδρομή

3.2 Νευρώνας

Τα νευρωνικά δίκτυα είναι μια κατηγορία αλγορίθμων μηχανικής μάθησης που χρησιμοποιούνται για τη μοντελοποίηση πολύπλοκων μοτίβων σε σύνολα δεδομένων χρησιμοποιώντας πολλαπλά κρυφά επίπεδα και μη γραμμικές συναρτήσεις ενεργοποίησης. Ένα νευρωνικό δίκτυο δέχεται μια είσοδο, την περνά μέσα από πολλαπλά στρώματα κρυφών και προσδίδει μια πρόβλεψη που αντιπροσωπεύει τη συνδυασμένη είσοδο όλων των νευρώνων. Ο τρόπος με τον οποίο ο νευρώνας επεξεργάζεται τις εισόδους του xj δίνεται από την εξίσωση:[4]

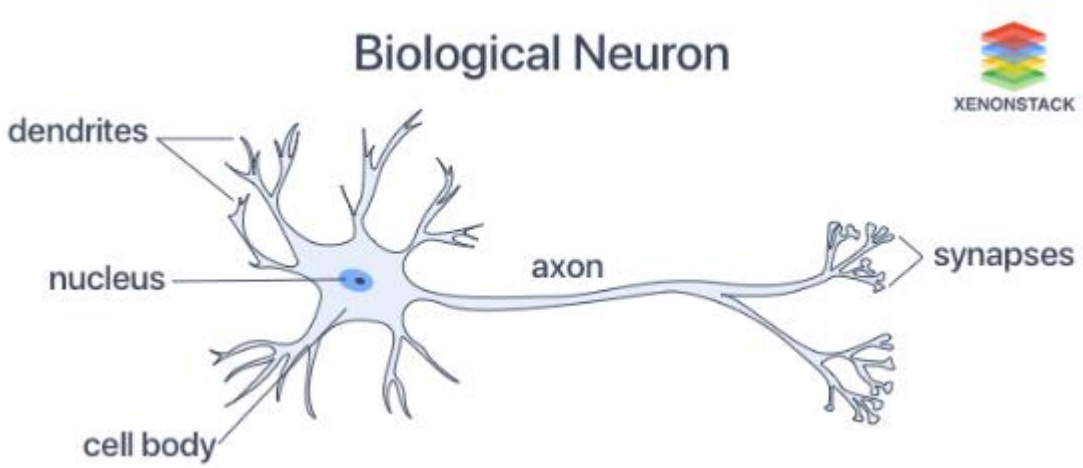
$$y = \Sigma_j (wjxj + b)$$

όπου wj ονομάζονται συναπτικά βάρη και bn παράμετρος, η οποία ονομάζεται πόλωση. Αυτές αποτελούν τις παραμέτρους του νευρώνα και καλούμαστε να τις υπολογίσουμε μέσω της διαδικασίας εκπαίδευσης.

3.2.1 Τεχνητός και Βιολογικός Νευρώνας

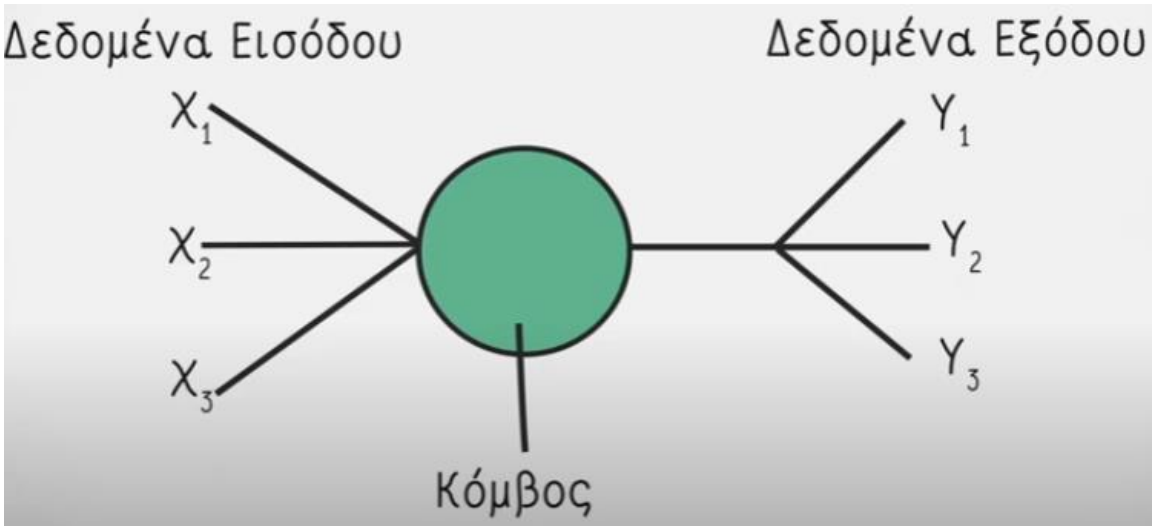
Υπάρχουν δισεκατομμύρια νευρώνες σε έναν ανθρώπινο εγκέφαλο. Στον ανθρώπινο εγκέφαλο, οι νευρώνες είναι δικτυωμένα νευρικά κύτταρα που επεξεργάζονται και στέλνουν ηλεκτρικά και χημικά μηνύματα. [14] Ένας βιολογικός νευρώνας αποτελείται από το κυτταρικό σώμα, όπου εκεί γίνονται οι απαραίτητοι υπολογισμοί, από τον πυρήνα, τους δενδρίτες, οι οποίοι λαμβάνουν τα σήματα εισόδου στο κυτταρικό σώμα και τις νευραξονικές απολήξεις που μεταφέρουν το επεξεργασμένο σήμα σε άλλους νευρώνες μέσω του νευράξονα Axon). Οι βιολογικοί νευρώνες συνδέονται μεταξύ τους και κάθε σύνδεση έχει διαφορετική βαρύτητα, δηλαδή το σήμα που μεταφέρεται, θα διαλέξει μια συγκεκριμένη πορεία από νευρώνες κάθε φορά βάση κάθε προτεραιότητας. [19]Ειδικότερα, πολλαπλά σήματα φτάνουν στους δενδρίτες και στη συνέχεια ενσωματώνονται στο σώμα του κυττάρου και, εάν

το συσσωρευμένο σήμα υπερβεί ένα ορισμένο όριο, παράγεται ένα σήμα εξόδου που θα μεταδοθεί από τον άξονα.[14]

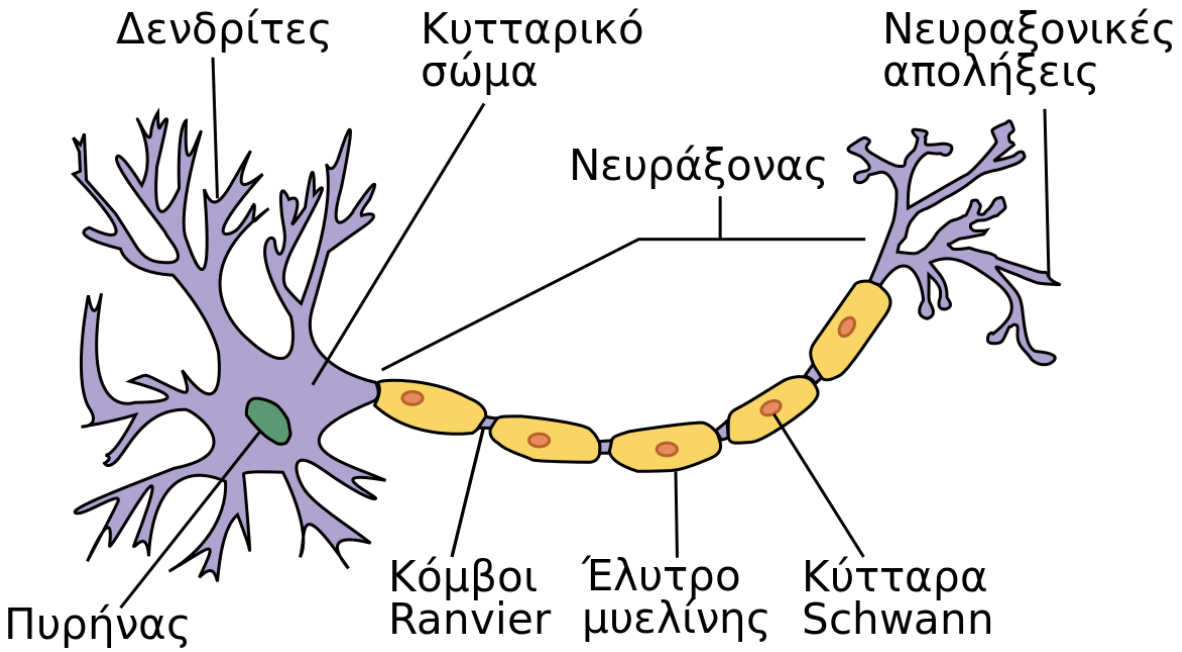


Εικόνα 2: Βιολογικός Νευρώνας

Αντίστοιχα, ένας τεχνητός νευρώνας είναι μια μαθηματική συνάρτηση που βασίζεται σε ένα μοντέλο βιολογικών νευρώνων που επεξεργάζεται τις εισόδους μέσω μιας μη γραμμικής συνάρτησης για την παραγωγή εξόδου.[14] Συγκεκριμένα, περιλαμβάνει έναν κόμβο, ο οποίος λαμβάνει δεδομένα στην είσοδο και τα επεξεργάζεται και μία ή πολλές εξόδους που οδηγούν τα νέα δεδομένα στους επόμενους κόμβους.[18]



Εικόνα 3: Κόμβος



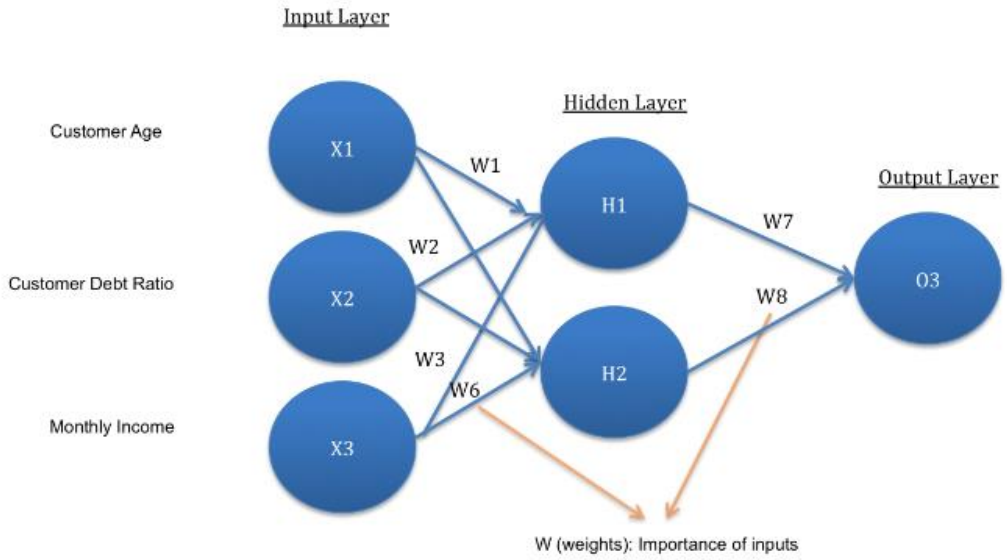
Εικόνα 4: Τεχνητός Νευρώνας

Τα χαρακτηριστικά ενός Τεχνητού Νευρώνα είναι αρκετά. Πρώτα από όλα, βασίζεται στη συμπεριφορά των βιολογικών νευρώνων. Ένα δεύτερο χαρακτηριστικό είναι πως αποτελεί ένα βασικό δομικό στοιχείο ενός τεχνητού νευρωνικού δικτύου. Ακόμη, σε μία ή περισσότερες εισόδους εφαρμόζονται ξεχωριστά βάρη καθώς και για την παραγωγή εξόδου, οι είσοδοι προστίθενται και στη συνέχεια αποστέλλονται μέσω μιας μη γραμμικής συνάρτησης. Επιπλέον, κάθε νευρώνας έχει μια εσωτερική κατάσταση γνωστή ως σήμα ενεργοποίησης και έναν συνδετικό κρίκο που τον συνδέει με κάθε άλλο νευρώνα. Σημαντικό, αποτελεί το γεγονός ότι ένας είναι μια μαθηματική συνάρτηση που βασίζεται σε ένα μοντέλο βιολογικών νευρώνων που επεξεργάζεται τις εισόδους μέσω μιας μη γραμμικής συνάρτησης για την παραγωγή εξόδου. Κάθε νευρώνας λαμβάνει εισόδους, τις ζυγίζει ξεχωριστά, τις προσθέτει μαζί και το κάνει αυτό για κάθε είσοδο ξεχωριστά.

Εξίσου σημαντικές είναι και οι διαφορές μεταξύ ενός βιολογικού και τεχνητού νευρώνα. Ειδικότερα, ένας βιολογικός νευρώνας δεν είναι σε θέση να θυμάται κάθε φορά το λάθος του ενώ ο τεχνητός βασίζεται σε κάθε λάθος του με σκοπό να βελτιώνεται κάθε φορά.[14] Γενικά, ο Τεχνητός νευρώνας μπορεί να αναλύσει πολύ σύνθετα προβλήματα, όπου συχνά δεν υπάρχουν προφανή συσχετισμοί στα δεδομένα.[18]

3.2.2 Δομή Τεχνητού Νευρωνικού Δικτύου

Η δομή ενός Τεχνητού Νευρωνικού Δικτύου περιλαμβάνει ένα επίπεδο εισόδου, ένα επίπεδο με κόμβους (κρυφό επίπεδο) και το επίπεδο εξόδου, δηλαδή τα αποτελέσματα. Όπως οι βιολογικοί, έτσι και οι τεχνητοί νευρώνες ενώνονται με συνδέσεις με διαφορετικής βαρύτητας.[18]

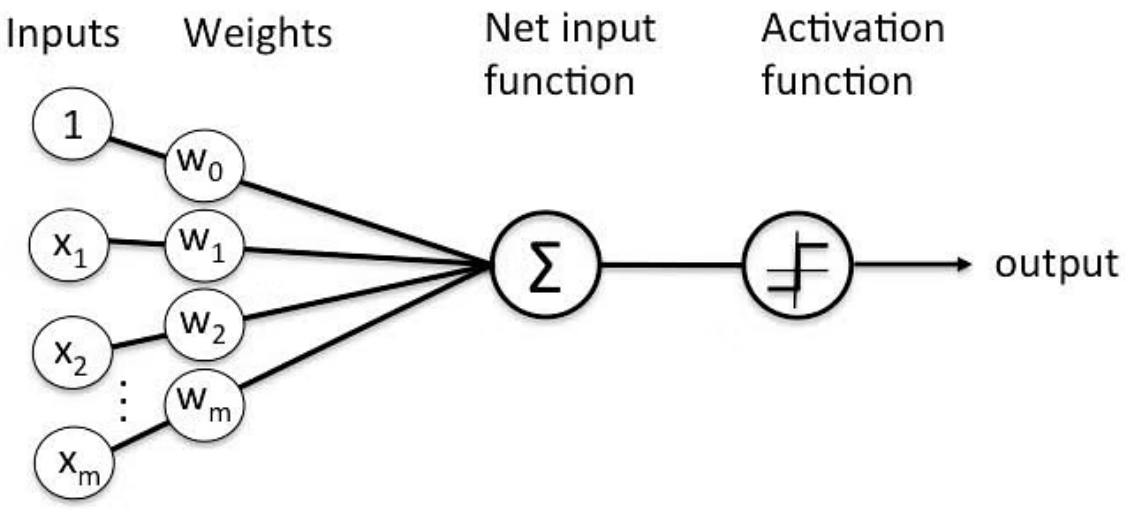


Εικόνα 5: Δομή Τεχνητού Νευρωνικού Δικτύου

Για παράδειγμα, μπορεί να υλοποιηθεί ένα σύστημα με σκοπό να μπορεί να ξεχωρίσει τις εικόνες με γάτες από ποντίκια. Στα επίπεδο εισόδου, μπορεί να υπάρξει ως χαρακτηριστικό x_1 τα πόδια και ως χαρακτηριστικό x_2 η ουρά. Ρωτάμε κάθε φορά το σύστημα, ποιο ζώο πιστεύει ότι είναι στη φωτογραφία με σκοπό να εκπαιδευτεί. Στην αρχή, ένα τεχνητό νευρωνικό δίκτυο απαντάει τυχαία γιατί βασίζεται σε τυχαία βάρη και τυχαίες τιμές των νευρώνων. Μετρώντας την απόκλιση από τον στόχο, ρυθμίζει κάθε φορά τις τιμές του δικτύου και υπολογίζει ξανά το αποτέλεσμα. Η διαδικασία αυτή επαναλαμβάνεται πολλές φορές μέχρις ότου το αποτέλεσμα να συγκλίνει στη σωστή τιμή. Σε αντίθεση με ένα βιολογικό νευρωνικό δίκτυο, το νευρωνικό μαθαίνει από τα λάθη του και τα θυμάται. Άρα, το Τεχνητό Νευρωνικό Δίκτυο είναι ένα μοντέλο μηχανικής μάθησης με ενίσχυση, το οποίο μπορεί να χρησιμοποιηθεί για ένα μεγάλο εύρος προβλημάτων, όπως ταξινόμηση, ομαδοποίηση και πρόβλεψη [18].

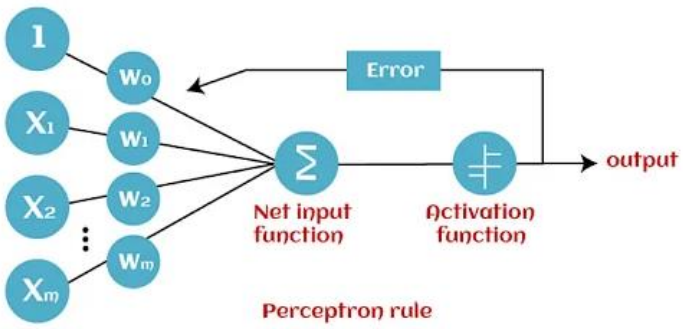
3.2.3 Perceptron

Το Perceptron είναι ο όρος που χρησιμοποιείται πιο συχνά στην τεχνητή νοημοσύνη και τη μηχανική μάθηση (AIML). Το Perceptron είναι ένας τύπος σύνδεσης του νευρωνικού δικτύου που χρησιμοποιεί τη τεχνητή νοημοσύνη στα δεδομένα εισόδου και στους υπολογισμούς για την παρακολούθηση χαρακτηριστικών. Ειδικότερα, συνδέεται με τους τεχνητούς νευρώνες χρησιμοποιώντας απλές λογικές πύλες με δυαδικές εξόδους. Σε σύγκριση με έναν πραγματικό νευρώνα, ένας τεχνητός νευρώνας επικαλείται τη μαθηματική συνάρτηση και έχει κόμβους, εισόδους, βάρη και εξόδους που είναι ισοδύναμες με τον κυτταρικό πυρήνα, τους δενδρίτες, τη σύναψη και τον άξονα, αντίστοιχα [14]. Το perceptron είναι ένας απλός αλγόριθμος που προορίζεται για την εκτέλεση δυαδικής ταξινόμησης. δηλαδή προβλέπει εάν η εισροή ανήκει σε μια συγκεκριμένη κατηγορία ενδιαφέροντος ή όχι. Για παράδειγμα, εάν κάτι είναι γάτα ή όχι [22].



Εικόνα 6: Αρχιτεκτονική Νευρώνα

Το Perceptron θεωρείται ένας νευρωνικός σύνδεσμος ενός στρώματος με τέσσερις κύριες παραμέτρους, τις τιμές εισόδου, τα βάρη, τη πόλωση (bias), το καθαρό άθροισμα και τη συνάρτηση ενεργοποίησης. Το μοντέλο perceptron ξεκινά με τον πολλαπλασιασμό όλων των τιμών εισόδου και των βαρών τους και στη συνέχεια προσθέτει αυτές τις τιμές για να δημιουργήσει το σταθμισμένο άθροισμα. Για να υπάρξει το επιθυμητό αποτέλεσμα, αυτό το σταθμισμένο άθροισμα εφαρμόζεται επίσης στη συνάρτηση ενεργοποίησης f, τη συνάρτηση βήματος, ένα άλλο όνομα για αυτήν τη λειτουργία ενεργοποίησης.[14]



Εικόνα 7: Λειτουργία του Perceptron

Η λειτουργία βήματος ή η λειτουργία ενεργοποίησης είναι για τη διασφάλιση ότι η έξοδος αντιστοιχίζεται μεταξύ (0,1) ή (-1,1). Αξίζει να σημειωθεί πως η ισχύς ενός κόμβου καθορίζεται από το βάρος της εισόδου του και πως η καμπύλη της λειτουργίας ενεργοποίησης μπορεί επίσης να μετατοπιστεί προς τα πάνω ή προς τα κάτω χρησιμοποιώντας μια τιμή εισόδου.[14]

Το perceptron θεωρείται ως ένας γραμμικός ταξινομητής, δηλαδή είναι ένας αλγόριθμος που ταξινομεί την είσοδο διαχωρίζοντας δύο κατηγορίες με ευθεία γραμμή. [22]Ως πρώτο βήμα του Perceptron θεωρείται ο πολλαπλασιασμός όλων των τιμών εισόδων με των αντίστοιχων τιμών των βαρών και στη συνέχεια, με τη πρόσθεση αυτών με σκοπό να βρεθεί το σταθμισμένο άθροισμα.

$$\Sigma w_i \cdot x_i = x_1 w_1 + x_2 w_2 + x_3 w_3 + \cdots + x_4 w_4$$

Στην συνέχεια, ως βήμα δεύτερο, προστίθεται το bias σε αυτό το σταθμισμένο άθροισμα για τη βελτίωση της απόδοσης του μοντέλου. Το προαναφερθέν σταθμισμένο άθροισμα συνδυάζεται με μια συνάρτηση ενεργοποίησης για να παραχθεί η ακόλουθη έξοδος, η οποία μπορεί να είναι είτε δυαδική είτε συνεχής:

$$Y = f(\sum w_i \cdot x_i + b)$$

Ο κανόνας εκμάθησης Perceptron δηλώνει ότι ο αλγόριθμος θα μάθει αυτόματα τους βέλτιστους συντελεστές βάρους. Αυτά τα βάρη στη συνέχεια συνδυάζονται με τα χαρακτηριστικά εισόδου για να αποφασιστεί εάν ένας νευρώνας ενεργοποιείται ή όχι. Όταν το σύνολο των σημάτων εισόδου που λαμβάνει το perceptron φτάσει σε ένα προκαθορισμένο όριο, είτε εξάγει ένα σήμα είτε δεν επιστρέφει ένα. Αυτό μπορεί στη συνέχεια να χρησιμοποιηθεί για την πρόβλεψη της τάξης ενός δείγματος στο πλαίσιο της εποπτευόμενης μάθησης και ταξινόμησης.[14]Ένα perceptron παράγει μια ενιαία έξοδο που βασίζεται σε πολλές εισόδους πραγματικών τιμών σχηματίζοντας έναν γραμμικό συνδυασμό χρησιμοποιώντας τα βάρη εισόδου του (και μερικές φορές περνώντας την έξοδο μέσω μιας μη γραμμικής συνάρτησης ενεργοποίησης).[22]Στον κανόνα εκμάθησης Perceptron, η προβλεπόμενη έξοδος συγκρίνεται με τη γνωστή έξοδο. Εάν δεν ταιριάζει, το σφάλμα διαδίδεται προς τα πίσω για να επιτραπεί η προσαρμογή του βάρους.[14]

Το Perceptron είναι μια συνάρτηση που έχει μια είσοδο «x», η οποία, όπως προαναφέρθηκε, πολλαπλασιάζεται με τον μαθημένο συντελεστή βάρους. Παράγεται μια τιμή εξόδου "f(x)".

$$f(x) = \begin{cases} 1, & w \cdot x + b > 0 \\ 0, & \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases},$$

Όπου, w θεωρείται το διάνυσμα βαρών πραγματικής αξίας, b θεωρείται το bias(ένα στοιχείο που προσαρμόζει το όριο μακριά από την αρχή χωρίς καμία εξάρτηση από την τιμή εισόδου) και χείναι το διάνυσμα εισαγόμενων τιμών x

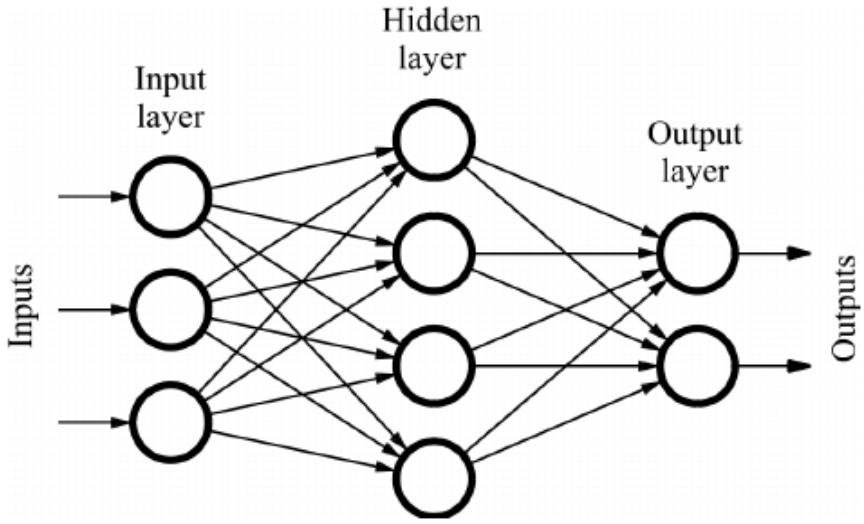
Το Perceptron χρησιμοποιείται και σε μονό και σε πολυστρωμικό στρώμα. Ο κύριος στόχος του μοντέλου perceptron μονής στρώσης είναι να αναλύσει τα γραμμικά διαχωρίσιμα αντικείμενα με δυαδικά αποτελέσματα και να μάθει μόνο γραμμικά διαχωρίσιμα μοτίβα ενώ στο πολυστρωμικό (Multilayer), το μοντέλο perceptron πολλαπλών επιπέδων περιέχει πολλαπλά κρυφά στρώματα και μπορεί να μάθει για δύο ή περισσότερα στρώματα που έχουν μεγαλύτερη επεξεργαστική ισχύ.

Συμπερασματικά, χρησιμοποιώντας το Perceptron, υπάρχουν αρκετά οφέλη. Για παράδειγμα, ένα πολυστρωματικό μοντέλο perceptron μπορεί να λύσει πολύπλοκα μη γραμμικά προβλήματα και μάλιστα, λειτουργεί καλά τόσο με μικρά όσο και με μεγάλα δεδομένα εισόδου. Επίσης, βοηθά στην απόκτηση γρήγορων προβλέψεων μετά την προπόνηση, έχοντας την ίδια αναλογία ακρίβειας με μεγάλα και μικρά δεδομένα.[14]

Παρά ταύτα, υπάρχουν και μειονεκτήματα της χρήσης του Perceptron. Το πολυεπίπεδο μοντέλο perceptron έχει το μειονέκτημα ότι οι υπολογισμοί είναι δύσκολοι και χρονοβόροι. Επίσης, είναι δύσκολη η πρόβλεψη για το πόσο η εξαρτημένη μεταβλητή επηρεάζει κάθε ανεξάρτητη μεταβλητή.[14]

3.3 Feedforward Networks – Δίκτυα τροφοδοσίας

Ένα νευρωνικό δίκτυο Feed Forward είναι ένα τεχνητό νευρωνικό δίκτυο στο οποίο οι συνδέσεις μεταξύ των κόμβων δεν σχηματίζουν κύκλο. Το αντίθετο από ένα νευρωνικό δίκτυο τροφοδοσίας είναι ένα επαναλαμβανόμενο νευρωνικό δίκτυο, στο οποίο πραγματοποιούνται κύκλοι ορισμένων οδών. Το μοντέλο προώθησης τροφοδοσίας είναι η απλούστερη μορφή νευρωνικού δικτύου καθώς οι πληροφορίες επεξεργάζονται μόνο προς μία κατεύθυνση. Ενώ τα δεδομένα μπορεί να περάσουν από πολλούς κρυφούς κόμβους, κινούνται πάντα προς μια κατεύθυνση και ποτέ προς τα πίσω. [20]



Εικόνα 8: Inputs and Outputs

Ένα Νευρωνικό Δίκτυο Τροφοδοσίας εμφανίζεται συνήθως στην απλούστερη μορφή του ως ένα μονό στρώμα perceptron. Όπως προαναφέρθηκε, η σειρά από εισόδους που εισέρχονται στο στρώμα, πολλαπλασιάζονται με τα βάρη και μετέπειτα οι σταθμισμένες τιμές εισόδου αθροίζονται για να παραχθεί ένα σύνολο. Η τιμή που παράγεται είναι συχνά 1 και αν το άθροισμα των τιμών είναι κάτω από το όριο, η τιμή εξόδου είναι -1 και μάλιστα το όριο συνήθως ορίζεται στο μηδέν.

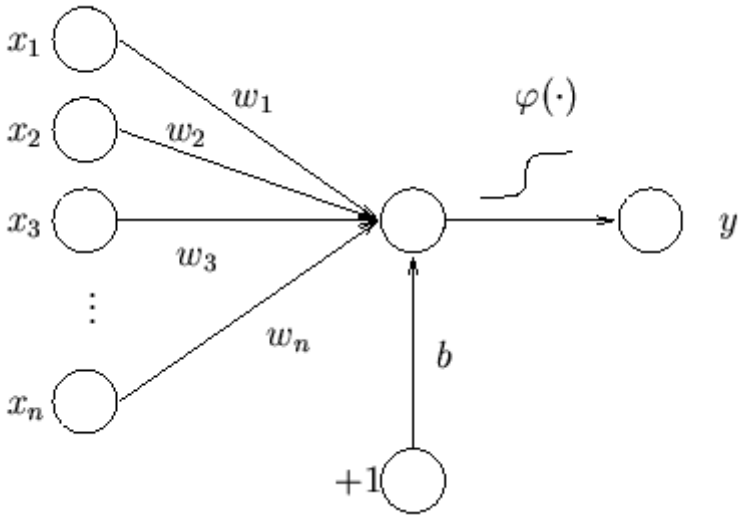
Στις εργασίες ταξινόμησης, το μονό στρώμα perceptron είναι ένα κρίσιμο μοντέλο νευρωνικού δικτύου τροφοδοσίας. Τα μονοστρωματικά perceptron μπορούν επίσης να περιέχουν ορισμένα χαρακτηριστικά της τεχνητής νοημοσύνης. Το νευρωνικό δίκτυο μπορεί να συγκρίνει τις εξόδους των κόμβων του με τις επιθυμητές τιμές χρησιμοποιώντας μια ιδιότητα γνωστή ως κανόνας δέλτα, η οποία επιτρέπει στο δίκτυο να εκπαιδεύει τα βάρη του για να δημιουργήσει τιμές εξόδου που είναι πιο ακριβείς. Αυτή η διαδικασία μάθησης και εκπαίδευσης έχει ως αποτέλεσμα μια βαθμιδωτή κάθοδο. Αν και η διαδικασία ενημέρωσης βαρών σε πολυστρωματικά perceptrons είναι σχεδόν συγκρίσιμη, είναι πιο τυπικά γνωστή ως back-propagation. Σε αυτές τις περιπτώσεις, τα κρυφά επίπεδα του δικτύου αλλάζουν το καθένα σύμφωνα με τις τιμές εξόδου που παράγονται από το ανώτερο επίπεδο.

3.3.1 Multilayer Perceptrons

Ένα πολυστρωματικό perceptron (MLP) είναι ένα βαθύ, τεχνητό νευρωνικό δίκτυο. Αποτελείται από πολλά στρώματα νευρώνων που το καθένα αποτελείται από ένα επίπεδο εισόδου για τη λήψη του σήματος, ένα επίπεδο εξόδου που λαμβάνει μια απόφαση ή πρόβλεψη σχετικά με την είσοδο, και μεταξύ αυτών των δύο, έναν αυθαίρετο αριθμό κρυφών επιπέδων που είναι η πραγματική υπολογιστική μηχανή του MLP.] Σημαντικό αποτελεί το γεγονός πως ένα πολυστρωματικό μοντέλο perceptron μπορεί να επεξεργαστεί τόσο γραμμικά όσο και μη γραμμικά μοτίβα και έχει μεγαλύτερη ικανότητα επεξεργασίας. Επιπλέον, μπορεί να χρησιμοποιηθεί ως λογικές πύλες όπως AND, OR, XOR, XNOR και NOR [14]. Τέλος, τα MLP με ένα κρυφό στρώμα είναι ικανά να προσεγγίσουν οποιαδήποτε συνεχή συνάρτηση [23][22].

Πολλά ζητήματα εποπτευόμενης μάθησης επιλύονται με πολυστρωματικά perceptrons. Εξασκούνται σε ένα σύνολο ζευγών εισόδου-εξόδου και μαθαίνουν να μοντελοποιούν τις συσχετίσεις (ή τις εξαρτήσεις) μεταξύ αυτών των εισόδων και εξόδων. Η εκπαίδευση περιλαμβάνει την αλλαγή των παραμέτρων του μοντέλου, ή των βαρών και των μεροληψιών του, για τη μείωση της ανακρίβειας. Το ίδιο το λάθος μπορεί να αξιολογηθεί με διάφορους τρόπους, συμπεριλαμβανομένου του ριζικού μέσου τετραγώνου του σφάλματος. Ο αλγόριθμος backpropagation χρησιμοποιείται για να γίνουν αυτές οι τροποποιήσεις ζύγισης και μεροληψίας σε σχέση με μια συνάρτηση σφάλματος και για να λυθεί το πρόβλημα εποπτευόμενης μάθησης [23].

Το αρχικό perceptron του Rosenblatt χρησιμοποιούσε μια συνάρτηση βήματος Heaviside ως συνάρτηση ενεργοποίησης φ. Σήμερα, και ειδικά σε δίκτυα πολλαπλών επιπέδων, η συνάρτηση ενεργοποίησης υπερβολικής εφαπτομένης $\tanh(x)$ επιλέγεται συχνά, με τύπο $\frac{1}{1+e^{-x}}$. Αυτές οι συναρτήσεις χρησιμοποιούνται επειδή είναι μαθηματικά βολικές και είναι κοντά στη γραμμική κοντινή αρχή, αλλά γρήγορα διαποτίζονται από αυτήν. Τα δίκτυα MLP μπορούν έτσι να προσομοιώνουν αποτελεσματικά τόσο πολύ όσο και ασθενώς μη γραμμικές αντιστοιχίσεις.

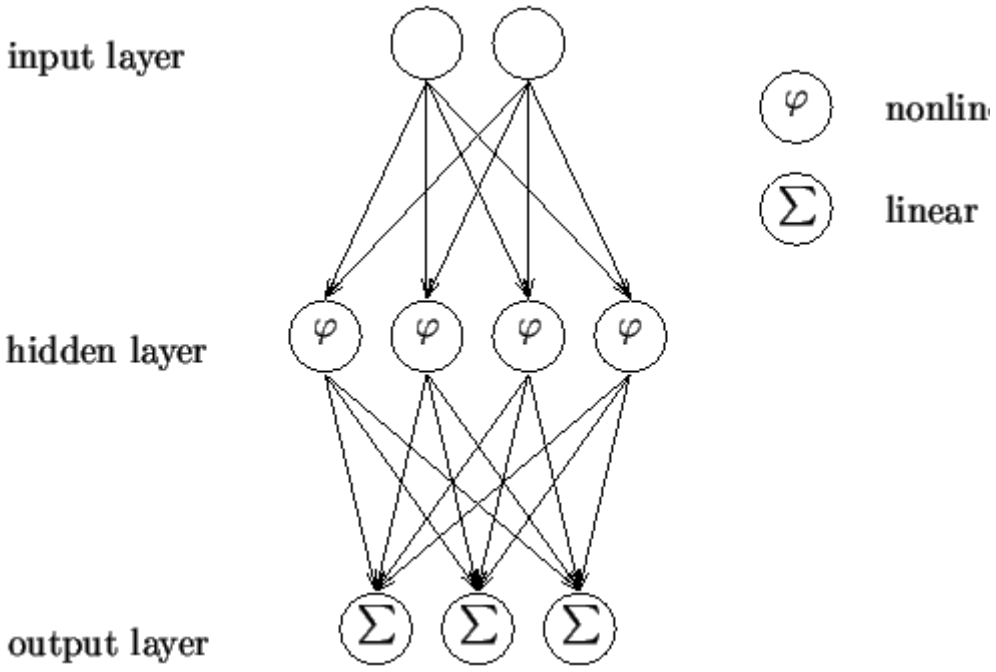


Εικόνα 9: Γράφημα ροής σήματος ενός MLP

Στο πολυστρωμικό στρώμα, υπάρχουν δύο στάδια, το μπροστινό πέρασμα (Forward Stage) και το πίσω πέρασμα (Backward Stage). Στο μπροστινό πέρασμα, η ροή του σήματος μετακινείται από το στρώμα

εισόδου μέσω των κρυφών στρωμάτων στο επίπεδο. Αντίθετα, στο πίσω πέρασμα, οι τιμές βάρους και bias προσαρμόζονται σύμφωνα με τις προδιαγραφές του μοντέλου. Τα διάφορα βάρη και προκαταλήψεις διαδίδονται πίσω μέσω του MLP. Αυτή η πράξη διαφοροποίησης μας δίνει μια κλίση ή ένα τοπίο σφάλματος, κατά μήκος του οποίου οι παράμετροι μπορούν να προσαρμοστούν καθώς μετακινούν το MLP ένα βήμα πιο κοντά στο ελάχιστο σφάλμα. Αυτό μπορεί να γίνει με οποιονδήποτε αλγόριθμο βελτιστοποίησης που βασίζεται σε κλίση, όπως το στοχαστικό gradient descent. Το δίκτυο συνεχίζει να εκπαιδεύεται έως ότου το σφάλμα δεν μπορεί να μειωθεί. Αυτή η κατάσταση είναι γνωστή ως σύγκλιση [23].

Οι υπολογισμοί που εκτελούνται από ένα τέτοιο δίκτυο προώθησης με ένα μόνο κρυφό στρώμα με μη γραμμικές συναρτήσεις ενεργοποίησης και ένα γραμμικό επίπεδο εξόδου μπορούν να γραφούν μαθηματικά ως $x = f(s) = B\varphi\ (As + a) + b$(4.5), όπου s είναι ένα διάνυσμα εισόδων, x ένα διάνυσμα εξόδων, A ο πίνακας βαρών του πρώτου στρώματος, a το διάνυσμα πόλωσης (bias) του πρώτου στρώματος, B ο πίνακας βάρους, b το διάνυσμα πόλωσης του δεύτερου στρώματος και το φ υποδηλώνει μια μη γραμμικότητα βάσει στοιχείων. Η γενίκευση του μοντέλου σε πιο κρυφά επίπεδα είναι προφανής [23].

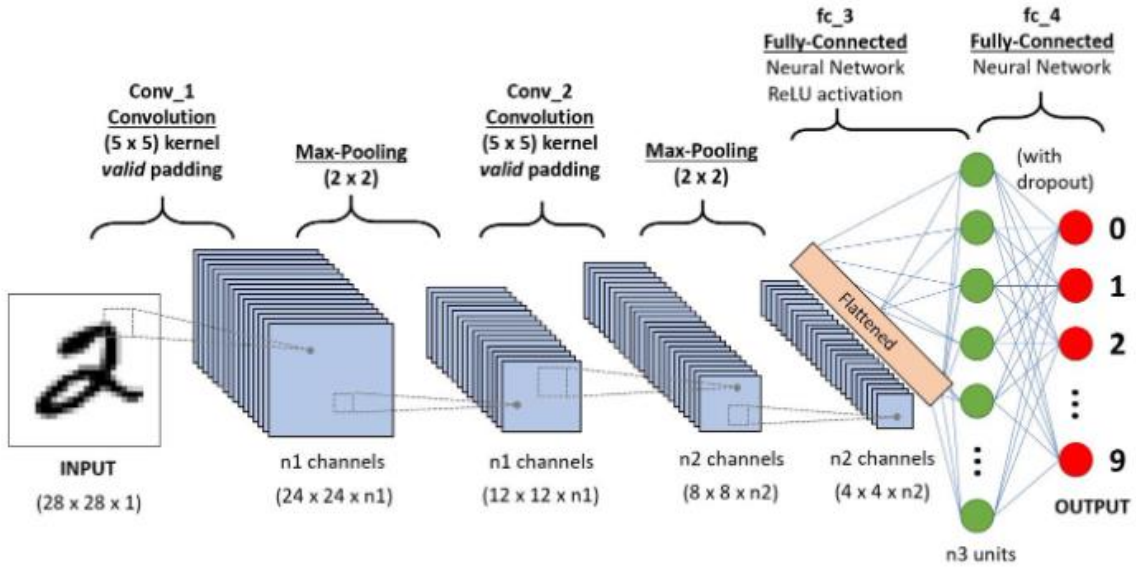


Εικόνα 10: Γράφημα ροής σήματος ενός MLP

3.3.2 CNN

Το Συνελικτικό Νευρωνικό Δίκτυο (Convolutional Neural Network) αποτελεί έναν αλγόριθμο, σύμφωνα με τον οποίο ένα μοντέλο είναι ικανό να αντιλαμβάνεται τον κόσμο όπως οι άνθρωποι. Για παράδειγμα, ένα μοντέλο, χρησιμοποιώντας αυτόν τον αλγόριθμο, βρίσκεται σε θέσει να αναγνωρίσει εικόνες και βίντεο, να αναλύσει και να ταξινομήσει μια εικόνα και να επεξεργαστεί τη φυσική γλώσσα. Συνοπτικά, το μοντέλο, χρησιμοποιώντας τον CNN αλγόριθμο, διαθέτει μηχανική, υπολογιστική ή τεχνητή όραση με σκοπό να δημιουργεί την αίσθηση της όρασης (computer vision) [24].

Το Συνελικτικό Νευρωνικό Δίκτυο (ConvNet/CNN) είναι μια μέθοδος Βαθιάς Μάθησης (Deep Learning) που μπορεί να λάβει μια εικόνα εισόδου, να αποδώσει σημασία (μαθησιακά βάρη και προκαταλήψεις) σε διάφορες πτυχές/αντικείμενα της εικόνας και να μπορεί να τα διακρίνει μεταξύ τους. Η προεπεξεργασία που απαιτείται σε ένα ConvNet είναι πολύ χαμηλότερη σε σύγκριση με άλλους αλγόριθμους ταξινόμησης. Ενώ στις πρωτόγονες μεθόδους τα φίλτρα κατασκευάζονται με το χέρι, με αρκετή εκπαίδευση, τα ConvNets έχουν τη δυνατότητα να μαθαίνουν αυτά τα φίλτρα/χαρακτηριστικά [24].



Εικόνα 11: Μια ακολουθία CNN για την ταξινόμηση χειρόγραφων ψηφίων

Η αρχιτεκτονική ενός ConvNet επηρεάστηκε από το πώς είναι οργανωμένος ο οπτικός φλοιός και είναι παρόμοιος με το δίκτυο συνδεσιμότητας των νευρώνων στον ανθρώπινο εγκέφαλο. Μόνο σε αυτή την περιορισμένη περιοχή του οπτικού πεδίου, που είναι γνωστή ως Δεκτικό Πεδίο, οι μεμονωμένοι νευρώνες αντιδρούν σε ερεθίσματα. Ολόκληρο το οπτικό πεδίο καλύπτεται από μια σειρά από τέτοια πεδία που επικαλύπτονται.[24]

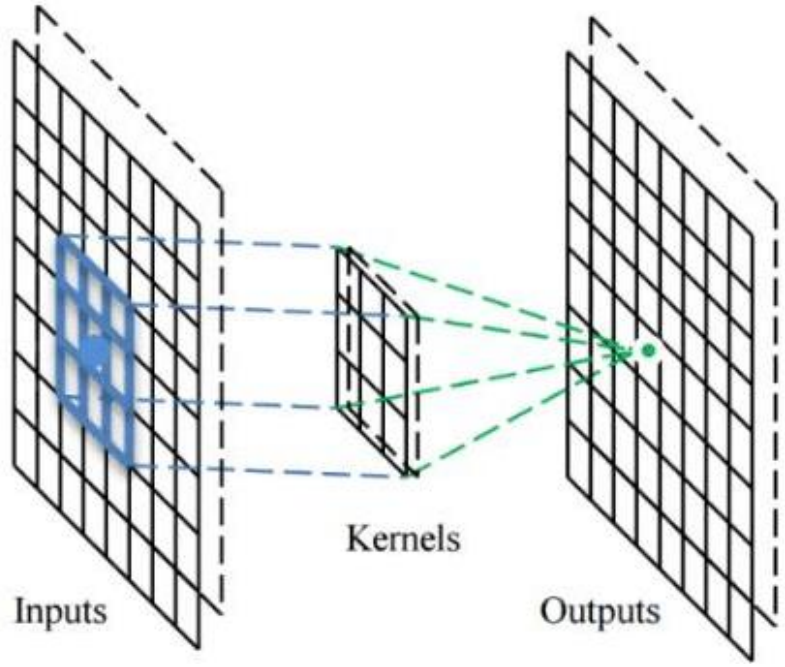
Το CNN χρησιμοποιείται για πολλά προβλήματα και συχνότερα για ταξινόμηση εικόνων, όπως είναι η ανίχνευση αυτοκινητοδρόμων σε δορυφορικές φωτογραφίες ή η κατηγοριοποίηση χειρόγραφων χαρακτήρων και αριθμών. Εκτός από αυτές τις πιο συνηθισμένες εργασίες, το CNN υπερέχει επίσης στην επεξεργασία σήματος και στην τμηματοποίηση εικόνας. Επίσης, αν και τα επαναλαμβανόμενα νευρωνικά δίκτυα (RNN) χρησιμοποιούνται συχνά για την επεξεργασία φυσικής γλώσσας (NLP), τα CNN έχουν χρησιμοποιηθεί για την κατανόηση στην αναγνώριση ομιλίας. Τέλος, Ένα CNN μπορεί επίσης να δημιουργηθεί χρησιμοποιώντας μια αρχιτεκτονική U-Net, η οποία είναι ουσιαστικά δύο σχεδόν κατοπτρικά CNN που συνδυάζονται για να δημιουργήσουν ένα CNN με αρχιτεκτονική σχήματος U. Τα U-net χρησιμοποιούνται για εργασίες όπως η τμηματοποίηση και η βελτίωση εικόνας όπου τα μεγέθη εξόδου και εισόδου πρέπει να ταιριάζουν [25].

Ο λόγος που δεν χρησιμοποιείται το Perceptron και χρησιμοποιείται το CNN είναι επειδή η προσέγγιση μπορεί να εκτελεί προβλέψεις κλάσεων με μέση βαθμολογία ακρίβειας για εξαιρετικά απλές δυαδικές

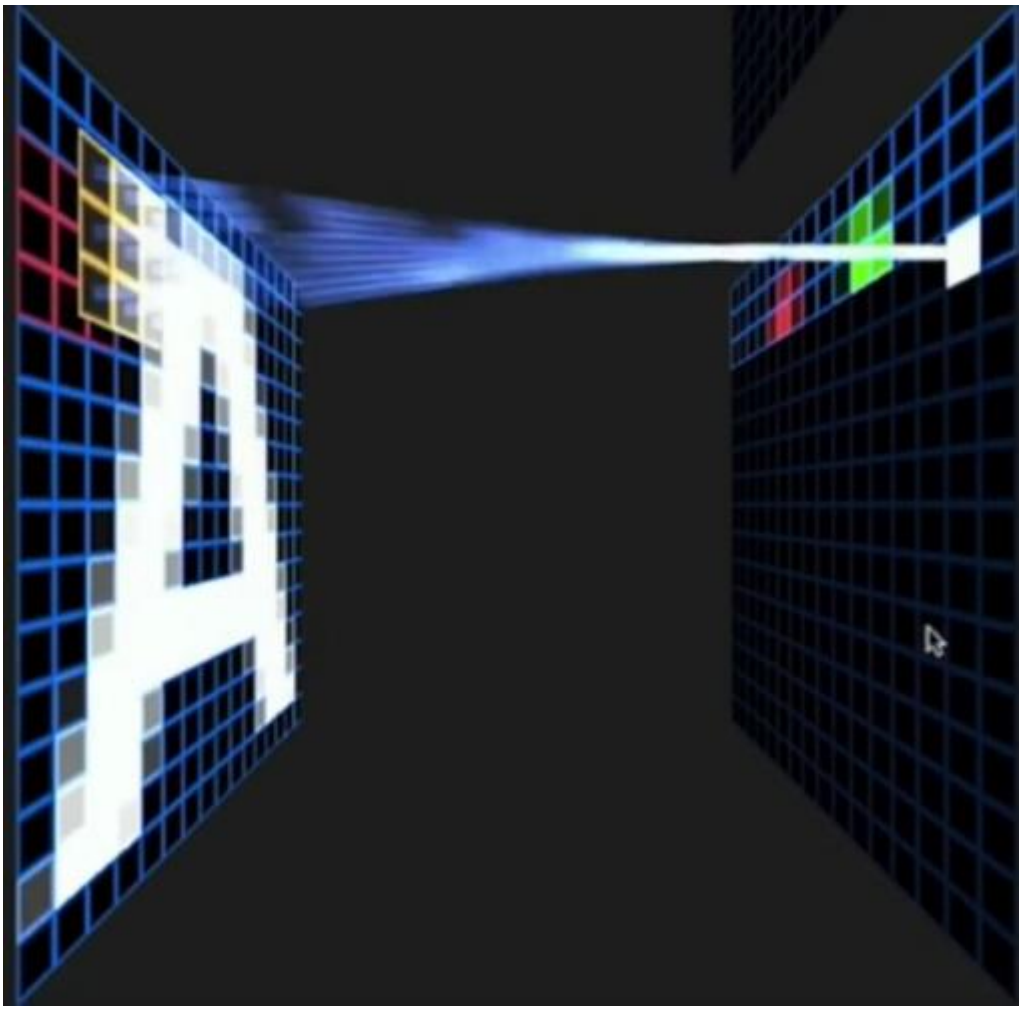
εικόνες, αλλά θα αποδίδει με μικρή έως καθόλου ακρίβεια για περίπλοκες εικόνες που έχουν εξαρτήσεις εικονοστοιχείων παντού.[24]

Μέσω της χρήσης σχετικών φίλτρων, ένα ConvNet μπορεί να αποτυπώσει αποτελεσματικά τις χωρικές και χρονικές εξαρτήσεις σε μια εικόνα. Επειδή υπάρχουν λιγότεροι παράγοντες που πρέπει να ληφθούν υπόψη και τα βάρη μπορούν να επαναχρησιμοποιηθούν, η αρχιτεκτονική παρέχει καλύτερη προσαρμογή στο σύνολο δεδομένων εικόνας. Με άλλα λόγια, το δίκτυο μπορεί να εκπαιδευτεί ώστε να κατανοεί καλύτερα το επίπεδο πολυπλοκότητας της εικόνας [24]. Για παράδειγμα, έστω ότι δίνεται μια φωτογραφία RGB (κόκκινο, πράσινο, μπλε). Ο ρόλος του ConvNet είναι να συμπυκνώνει τις εικόνες σε μια μορφή που είναι πιο εύκολη στην επεξεργασία, χωρίς να χάσει χαρακτηριστικά που είναι κρίσιμα για την επίτευξη καλής πρόβλεψης. Αυτό είναι σημαντικό για τη δημιουργία μιας αρχιτεκτονικής, η οποία βοηθάει όχι μόνο στην εκμάθηση χαρακτηριστικών, αλλά και στην επεκτασιμότητα τεράστιων συνόλων δεδομένων. Στην ουσία το CNN επικεντρώνεται στο να βρει ορισμένα χαρακτηριστικά που μπορεί να είναι πιο αποτελεσματικό παρά σε όλη την εικόνα [24].

Κάθε συνελικτικό στρώμα (convolutional layer) περιέχει μια σειρά από φίλτρα γνωστά ως συνελικτικοί πυρήνες. Ένας πίνακας ακεραίων αριθμών ίδιου μεγέθους με τον πυρήνα χρησιμεύει ως φίλτρο, το οποίο εφαρμόζεται σε ένα υποσύνολο των τιμών των εικονοστοιχείων εισόδου. Για απλότητα, κάθε εικονοστοιχείο πολλαπλασιάζεται με την αντίστοιχη τιμή στον πυρήνα προτού το αποτέλεσμα αθροιστεί για να αναπαραστήσει ένα κελί πλέγματος ή ένα εικονοστοιχείο, στον χάρτη του καναλιού/χαρακτηριστικού εξόδου. Αυτοί είναι γραμμικοί μετασχηματισμοί, κάθε συνέλιξη είναι ένας τύπος συγγενικής συνάρτησης.[25]



Εικόνα 12: CNN Input- Outputs



Εικόνα 13: Ο πυρήνας σαρώνει τις τιμές στον πίνακα εισόδου

Για τον χειρισμό των εικονοστοιχείων άκρων υπάρχουν διάφορες προσεγγίσεις. Για παράδειγμα, υπάρχουν προσεγγίσει για την απώλεια pixel στις άκρες, γέμισμα χρησιμοποιώντας pixel χωρίς τιμή, και επένδυση για αντανάκλαση. Η ιδανική μέθοδος είναι το reflection padding, το οποίο περιλαμβάνει την αντιγραφή εικονοστοιχείων από την άκρη της εικόνας και την προσθήκη τους στο εξωτερικό για να συμπληρώσει την ποσότητα των pixel που απαιτείται για τον συνελικτικό πυρήνα και για την επεξεργασία των εικονοστοιχείων της άκρης. Ένα ακόμη pixel πρέπει να προστεθεί γύρω από το εξωτερικό ενός πυρήνα 3x3 και τρία επιπλέον pixel πρέπει να αντανακλώνται γύρω από το εξωτερικό ενός πυρήνα 7x7. Η διάσταση διαιρείται στο μισό και στρογγυλοποιείται προς τα κάτω με τον αριθμό των pixel γύρω από κάθε πλευρά.[25]

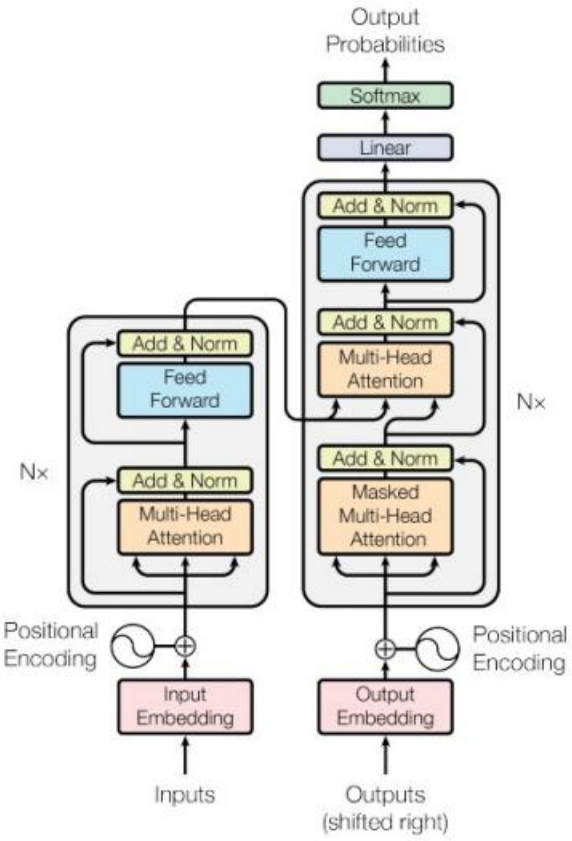
Στα μοντέλα του CNN, τα συνελικτικά επίπεδα περιέχουν μερικές φορές 16 ή ακόμα και 64 συνελικτικούς πυρήνες, που είναι πολύ περισσότεροι από τους τυπικούς τρεις. Αυτοί οι πολλοί πυρήνες συνέλιξης λειτουργούν ως ξεχωριστό φίλτρο για την παραγωγή ενός χάρτη καναλιού/χαρακτηριστικών που αντιπροσωπεύει μια διαφορετική έννοια. Οι πυρήνες θα μπορούσαν, για παράδειγμα, να φιλτράρουν τις επάνω άκρες, τις κάτω άκρες, τις διαγώνιες γραμμές κ.λπ. Αυτοί οι πυρήνες μπορεί να φιλτράρονται σε ζωικά χαρακτηριστικά όπως μάτια ή φτερά πουλιών σε πολύ βαθύτερα δίκτυα.

Έχοντας μεγαλύτερο αριθμό συνελικτικών πυρήνων, αυξάνονται και οι χάρτες καναλιών/χαρακτηριστικών και τα δεδομένα που προκύπτουν, γεγονός που αυξάνει τη χρήση της μνήμης.

3.3.3 Transformer Neural Network – Μετασχηματισμός Νευρωνικού Δικτύου

Λόγω της ικανότητας του μηχανισμού Attention να ανιχνεύει μοτίβα και συσχετίσεις μεταξύ όλων των επιμέρους στοιχείων μιας ακολουθίας, τείνει να χρησιμοποιείται όλο και περισσότερο σε προβλήματα φυσικής γλώσσας σε συνδυασμό με επαναλαμβανόμενα ή και συνελικτικά νευρωνικά δίκτυα που δείχνουν καλή απόδοση σε προβλήματα ακολουθίας. Ωστόσο, μοντέλα που συνδυάζουν επαναλαμβανόμενα δίκτυα και Attention έχουν δυσκολίες στην επεξεργασία μεγάλων ακολουθιών, καθώς η συσχέτιση μεταξύ των στοιχείων μιας ακολουθίας μειώνεται με την αύξηση της απόστασης μεταξύ τους. Για να αντιμετωπιστεί αυτό το πρόβλημα, χρησιμοποιήθηκαν μοντέλα που συνδυάζουν συνελικτικά και αναδρομικά επίπεδα, όπου εφαρμόζονται συνελίξεις στην ακολουθία εισόδου και στην συνέχεια το αποτέλεσμα τροφοδοτείται στο αναδρομικό επίπεδο. Σε αυτές τις συνθήκες, τα συνελικτικά στρώματα εξάγουν τα χαρακτηριστικά από κάθε σημείο της ακολουθίας διατηρώντας τη σειριακή δομή των δεδομένων εισόδου. Το βασικό μειονέκτημα αυτών των συστημάτων είναι πόσα επίπεδα χρειάζονται για να εξαχθούν σωστά οι διαδοχικές εξαρτήσεις. Λόγω των αυξανόμενων αναγκών επεξεργασίας τους, τέτοια μοντέλα συχνά είτε δεν είναι εφαρμόσιμα στην πράξη είτε έχουν χαμηλή ακρίβεια. [28]

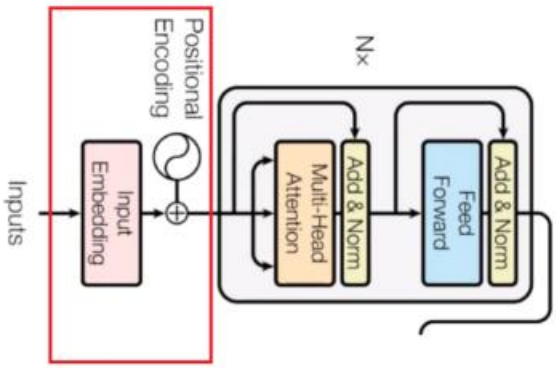
Η αρχιτεκτονική Transformer της Google, η οποία ασχολείται με την επεξεργασία ακολουθιών φυσικής γλώσσας, εισήχθη το 2017 σε μια προσπάθεια να αναπτυχθεί ένα μοντέλο που συνδυάζει τα πλεονεκτήματα των αναδρομικών και συνελικτικών δικτύων αποφεύγοντας τα μειονεκτήματά τους. Χωρίς τη χρήση συνελικτικών ή αναδρομικών επιπέδων, αυτός ο σχεδιασμός αποτελείται εξ ολοκλήρου από επίπεδα του μηχανισμού αυτοπροσοχής (Self Attention) και πλήρως συνδεδεμένα νευρωνικά δίκτυα. Ένας μηχανισμός Self Attention είναι ένα είδος μηχανισμού προσοχής που επιδιώκει να αναπαραστήσει μια ακολουθία εντοπίζοντας συσχετισμούς μεταξύ των συστατικών στοιχείων της. Αυτή η προσέγγιση χρησιμοποιείται με επιτυχία σε μια ποικιλία θεμάτων φυσικής γλώσσας, όπως η σύνοψη κειμένου, όπου ο στόχος είναι να εξαχθεί μια συμπαγής αναπαράσταση. Η προτεινόμενη αρχιτεκτονική στην προαναφερθείσα μελέτη τα πήγε καλύτερα από τα προηγούμενα μοντέλα και ήταν ένα ολοκαίνουργιο μοντέλο τελευταίας τεχνολογίας.[28]



Εικόνα 14: Σχήμα Αρχιτεκτονική Transformer

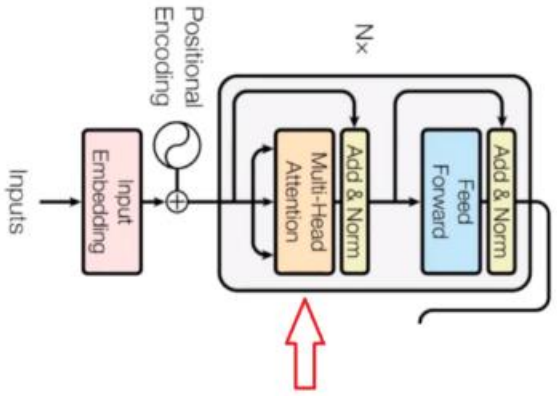
Οι υπολογιστές είναι γνωστό ότι λειτουργούν με αριθμούς, διανύσματα ή πίνακες και όχι με λέξεις. Για αυτό το λόγο πρέπει να μετατραπούν οι λέξεις σε διανύσματα. Για αυτό το λόγο δημιουργήθηκε η έννοια του Embedding Space, για να λύσει αυτό το πρόβλημα. Μοιάζει με έναν ανοιχτό χώρο ή λεξικό όπου λέξεις παρόμοιας σημασίας ομαδοποιούνται ή βρίσκονται η μία κοντά στην άλλη σε αυτόν τον χώρο. Αυτός ο χώρος ονομάζεται χώρος ενσωμάτωσης (Embedding Space) και κάθε λέξη, ανάλογα με τη σημασία της, αντιστοιχίζεται και αποδίδεται με μια συγκεκριμένη τιμή. Έτσι, μετατρέπονται οι λέξεις μας σε διανύσματα.[26]

Παρόλο που χρησιμοποιείται ο όρος Embedding Space για να μετατρέψει τις λέξεις σε διάνυσμα, δημιουργεί άλλα προβλήματα. Για παράδειγμα, ο μηχανισμός self-attention δεν έχει την ικανότητα διάκρισης της σειράς των σημείων μιας ακολουθίας και αυτό αποτελεί πρόβλημα επειδή σε κάθε πρόταση καθοριστικό ρόλο έχει η θέση κάθε λέξης μέσα σε αυτήν διότι προσδίδει διαφορετική σημασία. Έτσι, για να λυθεί αυτό το ζήτημα, χρησιμοποιείται η βοήθεια των κωδικοποιητών θέσης. Ο κωδικοποιητής (encoder) αποτελεί ένα διάνυσμα που δίνει το περιεχόμενο σύμφωνα με τη θέση της λέξης σε μια πρόταση.[26][28] Για να δημιουργηθεί το περιεχόμενο, πρώτα πρέπει να εντοπιστεί η λέξη, μετά να ενσωματωθεί(Embedding), μετέπειτα να ενσωματωθεί η θέση (Positional Embedding) και τέλος να οριστεί το διάνυσμα.[26]

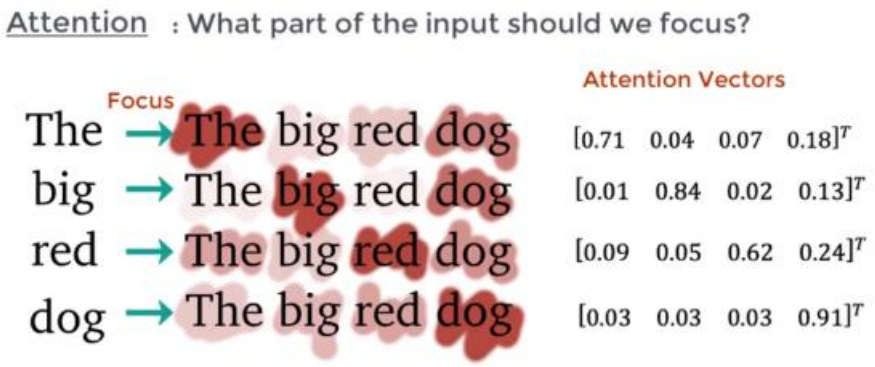


Εικόνα 15: Σχήμα Encoder Block

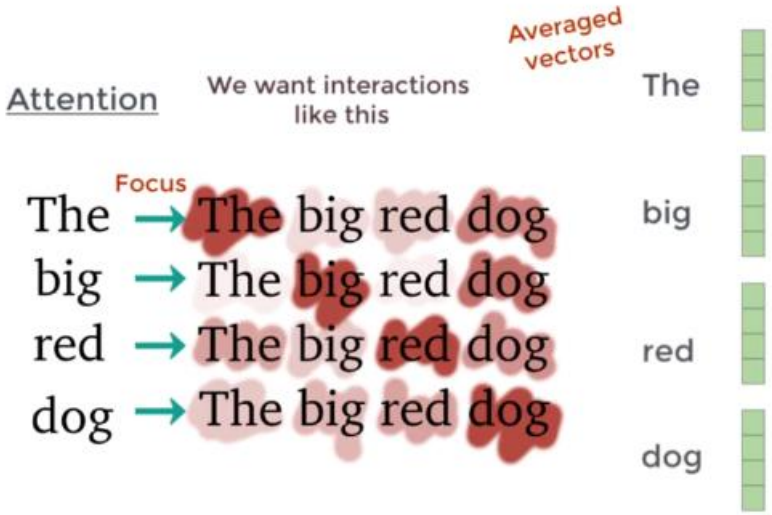
Η Προσοχή Πολλών Κεφαλών (Multi-Head Attention) εστιάζει στην πτυχή της σημασίας μιας λέξης σε σχέση με άλλες λέξεις της πρότασης και εμφανίζεται ως φορέας προσοχής. Η χρήση πολλαπλών διανυσμάτων προσοχής, ονομάζεται Multi-Head Attention Block. Ειδικότερα, ένα διάνυσμα προσοχής που καταγράφει τη σχέση συμφραζόμενων μεταξύ των λέξεων σε μια φράση μπορεί να κατασκευαστεί για κάθε λέξη. Το μόνο πρόβλημα που αντιμετωπίζει είναι ότι για κάθε λέξη βαραίνει πολύ περισσότερο την αξία της στην πρόταση. Έτσι, προσδιορίζουμε πολλαπλά διανύσματα προσοχής ανά λέξη και παίρνουμε έναν σταθμισμένο μέσο όρο, για να υπολογίσουμε το τελικό διάνυσμα προσοχής κάθε λέξης [26].



Εικόνα 16: Multi-Head Attention Part



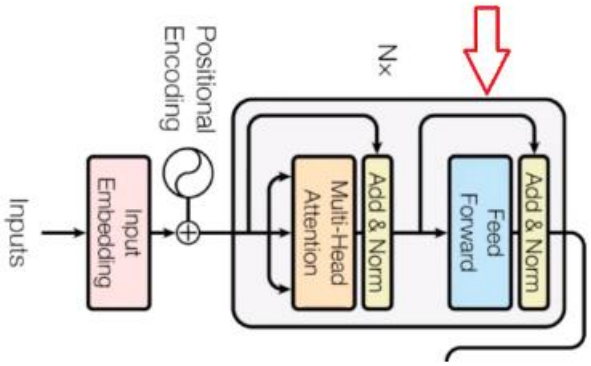
Εικόνα 17: Παράδειγμα: Το μοντέλο δεν ξέρει που να εστιάσει



Εικόνα 18: : Παράδειγμα: Το μοντέλο δεν ξέρει που να εστιάσει

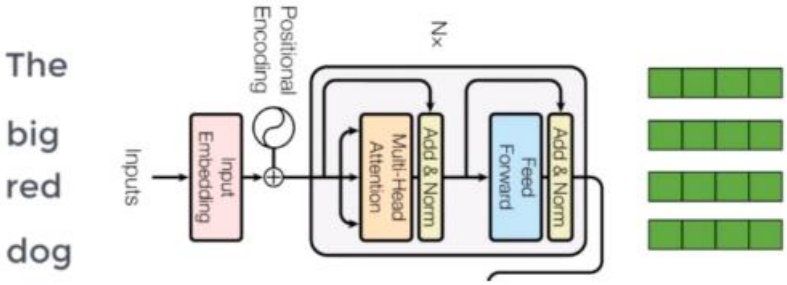
Στις παραπάνω εικόνες, εικόνα 17 και 18, παρατηρείται πως σε μια τόσο απλή πρόταση, το μοντέλο δεν ξέρει που να εστιάσει.

Το νευρωνικό δίκτυο Feed Forward είναι το επόμενο βήμα. Ο στόχος αυτού του απλού νευρωνικού δικτύου τροφοδοσίας, το οποίο εφαρμόζεται σε κάθε διάνυσμα προσοχής, είναι να αλλάξει τα διανύσματα προσοχής σε μια μορφή που μπορεί να υποβληθεί σε επεξεργασία από το επόμενο επίπεδο κωδικοποιητή ή αποκωδικοποιητή.[26]



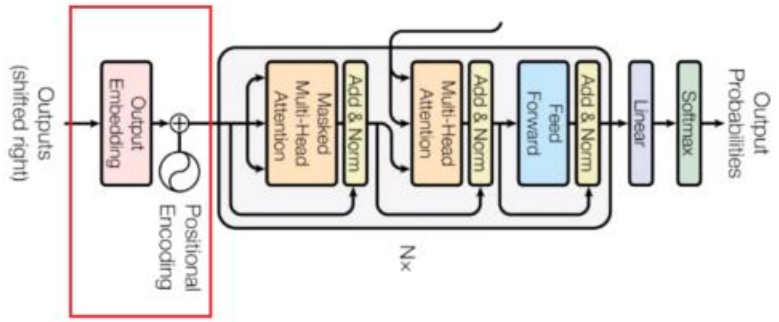
Εικόνα 19: Feed Forward Network

Το Feed Forward Network δέχεται διανύσματα προσοχής ένα κάθε φορά που το καθένα από αυτά τα διανύσματα προσοχής είναι ανεξάρτητο το ένα από το άλλο. Έτσι, η παραλληλοποίηση μπορεί να εφαρμοστεί εδώ, και αυτό κάνει τη διαφορά. Έτσι, πλέον μπορούν να περαστούν όλες οι λέξεις ταυτόχρονα στο encoder block και να πάρουμε το σύνολο των Κωδικοποιημένων Διανυσμάτων για κάθε λέξη ταυτόχρονα.[26]



Εικόνα 20: Encoder’s Output

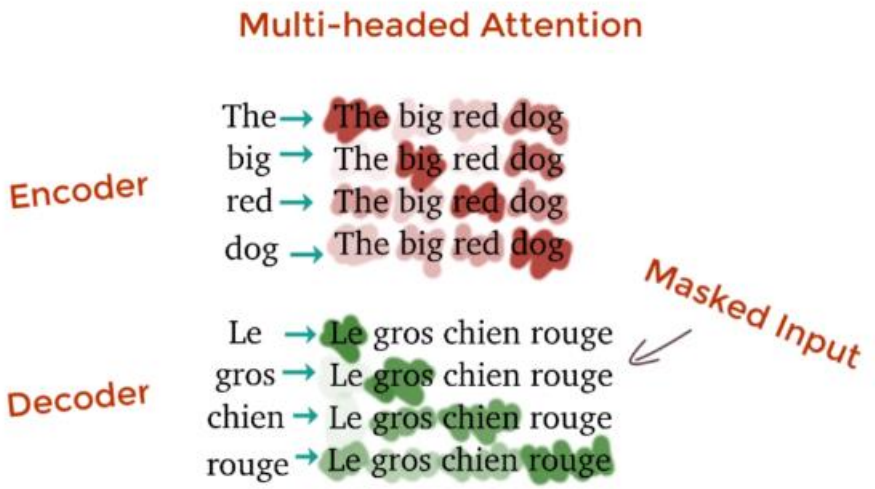
Όπως εκπαιδεύεται ένας μεταφραστής για να μεταφράσει τα αγγλικά στα γαλλικά, έτσι πρέπει να εκπαιδευτεί και το μοντέλο. Ειδικότερα, οι αγγλικές προτάσεις περνούν από το μπλοκ κωδικοποιητή (Encoder Block) και οι γαλλικές προτάσεις περνούν από το μπλοκ αποκωδικοποιητή (Decoder Block). Αξίζει να σημειωθεί πως η διαδικασία του Decoder Block, είναι ίδια με του encoder, δηλαδή στην αρχή βρισκόμαστε στο επίπεδο ενσωμάτωσης και στο τμήμα κωδικοποιητή θέσης που αλλάζει τις λέξεις σε αντίστοιχα διανύσματα.[26]



Εικόνα 21: Decoder Block

Το επόμενο βήμα θεωρείται το self-attention block, όπου δημιουργούνται διανύσματα προσοχής για κάθε λέξη σε γαλλικές προτάσεις για να αντιπροσωπεύουν πόσο σχετίζεται κάθε λέξη με κάθε λέξη της ίδιας πρότασης. Αυτή η διαδικασία ονομάζεται Masked Multi-Head Attention Part.

Ειδικότερα, για να μεταφραστούν οι λέξεις θα πρέπει πρώτα να εμφανιστεί ο αγγλικός όρος, ο οποίος θα μεταφραστεί στα γαλλικά χρησιμοποιώντας προηγούμενα ευρήματα, και στη συνέχεια θα συγκριθεί με την πραγματική γαλλική μετάφραση (την οποία τροφοδοτήσαμε στο μπλοκ αποκωδικοποιητή).Η τιμή του πίνακα θα ενημερωθεί μετά τη σύγκριση των δύο. Μετά από πολλές επαναλήψεις, θα μάθει το μοντέλο με αυτόν τον τρόπο. Σημαντικό αποτελεί το γεγονός, να αποκρυφθεί η γαλλική λέξη για να μπορέσει το μοντέλο να προβλέψει την λέξη, χρησιμοποιώντας προηγούμενα αποτελέσματα χωρίς να γνωρίζουμε τον πραγματικό μεταφρασμένο όρο, με σκοπό να εκπαιδευτεί.[26]



Εικόνα 22: Αυτό είναι το παράδειγμα με τα αγγλικά σε γαλλικά.

Αξίζει να σημειωθεί πως μπορούμε να χρησιμοποιήσουμε οποιαδήποτε λέξη από την αγγλική πρόταση, αλλά μπορούμε να πάρουμε μόνο την προηγούμενη λέξη της γαλλικής πρότασης για να μάθουμε. Έτσι, ενώ εκτελούμε την παραλληλοποίηση με τη λειτουργία του πίνακα βεβαιωνόμαστε ότι ο πίνακας θα

πρέπει να καλύπτει τις λέξεις που εμφανίζονται αργότερα μετατρέποντάς τες σε 0, έτσι ώστε το δίκτυο προσοχής να μην μπορεί να τις χρησιμοποιήσει. Στη συνέχεια, τα διανύσματα προσοχής που προκύπτουν από το προηγούμενο επίπεδο και τα διανύσματα από το μπλοκ κωδικοποιητή περνούν σε ένα άλλο μπλοκ προσοχής πολλαπλών κεφαλών, όπου τα αποτελέσματα από το μπλοκ κωδικοποιητή εμφανίζονται επίσης στην εικόνα. Στο διάγραμμα είναι επίσης ορατό ότι τα αποτελέσματα από τα μπλοκ κωδικοποιητή έρχονται εδώ.). Αυτός είναι ο λόγος για τον οποίο ονομάζεται Encoder-Decoder Attention Block.[26]

Για παράδειγμα, καθώς έχουμε ένα διάνυσμα για κάθε λέξη για κάθε αγγλική και γαλλική πρόταση. Αυτό το μπλοκ κάνει στην πραγματικότητα τη χαρτογράφηση αγγλικών και γαλλικών λέξεων και ανακαλύπτει τη σχέση μεταξύ τους. Λοιπόν, αυτό είναι το μέρος όπου συμβαίνει η κύρια αντιστοίχιση λέξεων από Αγγλικά σε Γαλλικά. Η έξοδος αυτού του μπλοκ είναι διανύσματα προσοχής για κάθε λέξη σε αγγλικές και γαλλικές προτάσεις. Κάθε διάνυσμα αντιπροσωπεύει τη σχέση με άλλες λέξεις και στις δύο γλώσσες.[26]

Τώρα που περνάμε κάθε διάνυσμα προσοχής σε μια μονάδα τροφοδοσίας, θα κάνει τα διανύσματα εξόδου να διαμορφωθούν σε κάτι που είναι εύκολα αποδεκτό από ένα άλλο μπλοκ αποκωδικοποιητή ή ένα γραμμικό στρώμα, το οποίο είναι ένα άλλο στρώμα τροφοδοσίας προς τα εμπρός (feed-forward layer) και χρησιμοποιείται για την επέκταση των διαστάσεων σε αριθμούς λέξεων στη γαλλική γλώσσα μετά τη μετάφραση. Η είσοδος υποβάλλεται τώρα σε επεξεργασία μέσω ενός στρώματος Softmax, το οποίο τη μετατρέπει σε κατανομή πιθανότητας που μπορεί να γίνει κατανοητή από τον άνθρωπο. Τέλος, μετά τη μετάφραση, ο τελικός όρος σχηματίζεται με τη μέγιστη πιθανότητα.[26]

Με άλλα λόγια, το Transformer, για κάθε λέξη, δημιουργεί αρχικά αρχικές αναπαραστάσεις ή ενσωματώσεις, οι οποίες αντιπροσωπεύονται από τους μη συμπληρωμένους κύκλους. Στη συνέχεια, χρησιμοποιώντας αυτοπροσοχή ,συγκεντρώνει πληροφορίες από όλες τις άλλες λέξεις, δημιουργώντας μια νέα αναπαράσταση ανά λέξη που ενημερώνεται από ολόκληρο το πλαίσιο, που αντιπροσωπεύεται από τις γεμάτες μπάλες. Αυτό το βήμα στη συνέχεια επαναλαμβάνεται πολλές φορές παράλληλα για όλες τις λέξεις, δημιουργώντας διαδοχικά νέες αναπαραστάσεις. Ο αποκωδικοποιητής λειτουργεί παρόμοια, αλλά παράγει μία λέξη τη φορά, από αριστερά προς τα δεξιά. Φροντίζει όχι μόνο τις άλλες λέξεις που δημιουργήθηκαν προηγουμένως αλλά και τις τελικές αναπαραστάσεις που δημιουργούνται από τον κωδικοποιητή. Έτσι, λοιπόν, λειτουργεί ο μετασχηματιστής και είναι πλέον η τελευταία λέξη της τεχνολογίας στο NLP. Τα αποτελέσματα είναι αρκετά ικανοποιητικά, χρησιμοποιώντας μηχανισμό αυτοπροσοχής και επίσης λύνει το θέμα της παραλληλοποίησης. Ακόμη και η Google χρησιμοποιεί το BERT που χρησιμοποιεί έναν μετασχηματιστή για την προεκπαίδευση μοντέλων για κοινές εφαρμογές NLP.

3.4 Αναδρομικά Δίκτυα - Recursive Neural Network

Τα βαθιά νευρωνικά δίκτυα περιλαμβάνουν αναδρομικά νευρωνικά δίκτυα, εφαρμόζοντας τον ίδιο αριθμό συνόλων βαρών σε δομημένες εισόδους, μπορεί να προβλεφθεί μια δομημένη πρόβλεψη. Με αυτόν τον τύπο επεξεργασίας, δημιουργείται ένα τυπικό βαθύ νευρωνικό δίκτυο, γνωστό ως αναδρομικό νευρωνικό δίκτυο. Αυτά τα δίκτυα είναι μη γραμμικής φύσης. Τα αναδρομικά δίκτυα είναι προσαρμοστικά μοντέλα που μπορούν να αποκτήσουν βαθιά δομημένη γνώση. Ως αποτέλεσμα, τα αναδρομικά νευρωνικά δίκτυα ανήκουν σε πολύπλοκες εγγενείς αλυσίδες.[29]

Ένα επαναλαμβανόμενο νευρωνικό δίκτυο (RNN) είναι ένας ειδικός τύπος τεχνητού νευρωνικού δικτύου προσαρμοσμένο να λειτουργεί για δεδομένα χρονοσειρών ή δεδομένα που περιλαμβάνουν

ακολουθίες. Τα συνηθισμένα νευρωνικά δίκτυα τροφοδοσίας προορίζονται μόνο για σημεία δεδομένων, τα οποία είναι ανεξάρτητα μεταξύ τους. Ωστόσο, εάν έχουμε δεδομένα σε μια ακολουθία τέτοια ώστε ένα σημείο δεδομένων να εξαρτάται από το προηγούμενο σημείο δεδομένων, πρέπει να τροποποιήσουμε το νευρωνικό δίκτυο ώστε να ενσωματωθούν οι εξαρτήσεις μεταξύ αυτών των σημείων δεδομένων. Τα RNN έχουν την έννοια της «μνήμης» που τους βοηθά να αποθηκεύουν τις καταστάσεις ή τις πληροφορίες προηγούμενων εισόδων για να δημιουργήσουν την επόμενη έξοδο της ακολουθίας.

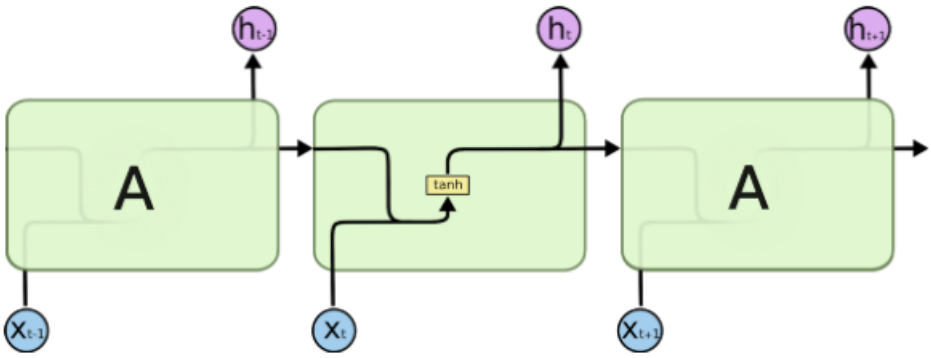
Τα αναδρομικά νευρωνικά δίκτυα ανήκουν σε έναν αρχιτεκτονικό τύπο που λειτουργεί καλύτερα με δομημένες εισόδους. Τα RNN εστιάζονται ειδικά σε άκυκλα γραφήματα και αυτό αποτελεί μια βαθιά δομή δέντρου. Τα αναδρομικά νευρωνικά δίκτυα χρησιμοποιούνται όταν είναι απαραίτητο να αναλυθεί ολόκληρη η πρόταση και έχει μια τοπολογία που μοιάζει με δέντρο ενώ τα RNN επιτρέπουν τη διακλάδωση των συνδέσεων και των ιεραρχικών δομών.[29]

Επιπλέον, χρησιμοποιούνται κυρίως τα αναδρομικά νευρωνικά δίκτυα για την πρόβλεψη δομημένων εξόδων. Γίνεται σε δομές εισόδου μεταβλητού μεγέθους και επίσης, διασχίζει μια δεδομένη δομή και αυτή με τοπολογική σειρά. Το κάνουν επίσης για βαθμωτές προβλέψεις αλλά εδώ πρέπει να σημειωθεί ότι το Αναδρομικό νευρωνικό δίκτυο απλώς δεν ανταποκρίνεται σε δομημένες εισόδους, αλλά λειτουργεί επίσης σε περιβάλλοντα. Κάθε χρονοσειρά αντιμετωπίζεται ανεξάρτητα. Αξίζει να σημειωθεί πως τα αρχικά RNN αναπτύχθηκαν για να μάθουν αναπαραστάσεις κατανεμημένων δεδομένων διαφορετικών δομικών δικτύων. Λογικές έννοιες, για παράδειγμα.[29]

Η χρήση των αναδρομικών δικτύων έχουν και πλεονεκτήματα και μειονεκτήματα. Πιο συγκεκριμένα, παρέχουν την δυνατότητα χειρισμού δεδομένων ακολουθίας, εισόδων διαφορετικού μήκους και τη δυνατότητα αποθήκευσης ή «απομνημόνευσης» ιστορικών πληροφοριών. Όμως, είναι αρκετά πολύ αργά και το δίκτυο δεν λαμβάνει υπόψη του μελλοντικές εισροές για τη λήψη αποφάσεων. Επίσης, παρουσιάζεται το πρόβλημα της εξαφανισμένης κλίσης, το οποίο εμποδίζει το δίκτυο να μάθει νέα βάρη, αναγκάζοντας τις διαβαθμίσεις που απαιτούνται για τον υπολογισμό της ενημέρωσης βάρους να πλησιάζουν δυνητικά εξαιρετικά κοντά στο μηδέν. Η σοβαρότητα αυτού του ζητήματος αυξάνεται με το βάθος του δικτύου αλλά πλέον λύνεται με του αλγορίθμους GRU και LSTM, τα οποία αναλύονται παρακάτω.[31]

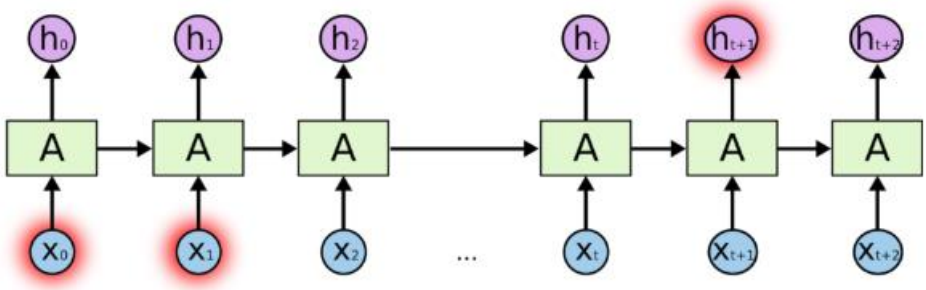
3.4.1 RNN

Τα RNN είναι τα νευρωνικά δίκτυα Feed Forward που αναπτύσσονται με την πάροδο του χρόνου. Σε αντίθεση με τα κανονικά νευρωνικά δίκτυα, τα RNN είναι σχεδιασμένα να λαμβάνουν μια σειρά εισόδων χωρίς προκαθορισμένο όριο μεγέθους και περιέχουν επαναλαμβανόμενες μονάδες στο κρυφό τους στρώμα, οι οποίες επιτρέπουν στον αλγόριθμο να επεξεργάζεται δεδομένα ακολουθίας, δηλαδή χρησιμοποιεί μια κρυφή κατάσταση για να θυμάται πληροφορίες. Αυτό γίνεται με την επαναλαμβανόμενη μετάβαση κρυφών καταστάσεων από προηγούμενα χρονικά βήματα και τον συνδυασμό τους με εισόδους της τρέχουσας [29][31]. Η "σειρά" όπως σε κάθε είσοδο αυτής της ακολουθίας έχει κάποια σχέση με τους γείτονές της ή έχει κάποια επιρροή σε αυτούς.[29]



Εικόνα 23: Αρχιτεκτονική RNN

Τα βασικά δίκτυα ανατροφοδότησης «θυμούνται» πράγματα, αλλά θυμούνται τα πράγματα που έμαθαν κατά τη διάρκεια της εκπαίδευσης. Ενώ τα RNN μαθαίνουν παρομοίως κατά την εκπαίδευση, επιπλέον, θυμούνται πράγματα που έμαθαν από προηγούμενες εισόδους κατά τη δημιουργία εξόδου.



Εικόνα 24: Εικόνα που απεικονίζει μακροπρόθεσμες εξαρτήσεις

Το RNNχρησιμοποιείται σε διαφορετικούς τύπος μοντέλων. Ειδικότερα, χρησιμοποιείται στο μοντέλο διανυσμάτων ακολουθίας (Vector-Sequence Models), όπου τα μοντέλα δέχονται διανύσματα σταθερού μεγέθους ως εισόδους και παράγουν διανύσματα αυθαίρετου μήκους ως εξόδους. Για παράδειγμα, στη λεζάντα εικόνας, η είσοδος είναι μια εικόνα και η έξοδος είναι μια περιγραφή της εισόδου. Ακόμη, χρησιμοποιείται στο μοντέλο ακολουθίας διανύσματος (Sequence-Vector Model), όπου από οποιοδήποτε διάνυσμα μεγέθους μπορεί να δημιουργηθεί ένα διάνυσμα με σταθερός μέγεθος. Για παράδειγμα, η κριτική μιας ταινίας βαθμολογείται ως θετική ή αρνητική χρησιμοποιώντας ένα διάνυσμα σταθερού μεγέθους σε μια ανάλυση συναισθήματος της ταινίας. Τέλος, το RNN χρησιμοποιείται από το μοντέλο ακολουθίας σε ακολουθία (Sequence-to-Sequence Model), όπου παίρνει μια ακολουθία εισόδου και την εξάγει ως μια άλλη ακολουθία με διάφορα μεγέθη. Για παράδειγμα, η εξέταση του ενδεχομένου να μεταφραστούν δεδομένα χρονοσειρών για προβλέψεις χρηματιστηρίου. .[26]

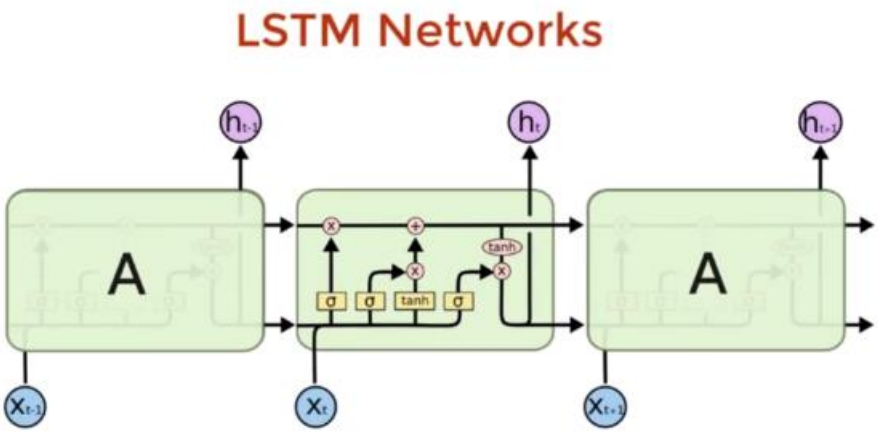
Η λειτουργία του RNN ξεκινάει έχοντας ως πρώτο βήμα την μετατροπή των ανεξάρτητων ενεργοποιήσεων σε εξαρτημένων κάνοντας όλα τα επίπεδα να περιέχουν τα ίδια βάρη και προβλέψεις. Αυτό, θα μειώσει την πολυπλοκότητα της αύξησης των παραμέτρων και της απομνημόνευσης καθενός από τα προηγούμενα αποτελέσματα δίνοντας την έξοδο ως είσοδο στο επόμενο κρυφό στρώμα. Έτσι, και τα τρία στρώματα θα είναι συνυφασμένα σε ένα ενιαίο επαναλαμβανόμενο στρώμα για να περιέχουν τα ίδια βάρη και προβλέψεις. Πιο συγκεκριμένα, κάθε φορά, δίνεται είσοδος στο δίκτυο και πρέπει να

υπολογιστεί η τρέχουσα κατάστασή του χρησιμοποιώντας το τρέχον σύνολο εισόδου και την προηγούμενη κατάσταση. Τώρα το τρέχον γίνεται το επόμενο βήμα. Όμως, μπορούν να γίνουν όσα χρονικά βήματα θελήσουμε για να συνδυαστούν τα δεδομένα από όλες τις προηγούμενες καταστάσεις. Μόλις ολοκληρωθούν όλα τα χρονικά βήματα, χρησιμοποιείται η τελική τρέχουσα κατάσταση για να υπολογιστεί η τελική έξοδο. Στο τέλος, καλούμαστε να συγκρίνουμε αυτήν την έξοδο με την πραγματική έξοδο και έτσι θα υπολογιστεί το σφάλμα. Αφού υπολογιστεί το σφάλμα, ενημερώνουμε τα βάρη για την εκπαίδευση του RNN.[30]

Παρόλα αυτά υπάρχουν και μειονεκτήματα της χρήσης του. Συγκεκριμένα, η προπόνηση του γίνεται με αρκετά αργό ρυθμό καθώς και οι μεγάλες ακολουθίες μπορεί να έχουν ως αποτέλεσμα την εξαφάνιση κλίσεων ή ζητήματα με μακροπρόθεσμες εξαρτήσεις. Δηλαδή, το μοντέλο θα έχει αδύναμη μνήμη, όταν πρόκειται για ανάκληση της προηγούμενης σύνδεσης. Για παράδειγμα, στην παρακάτω πρόταση « Το καλοκαίρι μου αρέσει να κολυμπάω στην __ » λείπει η λέξη «θάλασσα». Η απόσταση μεταξύ του ρήματος κολυμπάω και της προβλεπόμενης λέξης είναι μικρότερη, έτσι ώστε το RNN να μπορεί να την προβλέψει εύκολα. Όμως, εάν δοθεί στο μοντέλο η πρόταση «Μετακόμισα στη Γαλλία εδώ και δέκα χρόνια και έτσι αναγκαστικά έμαθα να μιλάω __ », το RNN δεν θα μπορεί να κατανοήσει ότι η λέξη που λείπει είναι «γαλλικά», επειδή η απόσταση μεταξύ της Γαλλίας και της προβλεπόμενης λέξης είναι μεγαλύτερη σε αυτήν την περίπτωση. Έτσι, δυστυχώς, καθώς αυτή η απόσταση μεγαλώνει, το RNN δεν μπορεί να λειτουργήσει, καθώς η μνήμη τους εξασθενεί με την απόσταση.

3.4.2 LSTM - Μακροπρόθεσμη Μνήμη

Ένας μοναδικός τύπος RNN που δημιουργήθηκε ειδικά για την εξαφάνιση προβλημάτων κλίσης αποτελεί η Μακροπρόθεσμη Μνήμη Βραχυπρόθεσμης (Long Short Term Memory). Έχει την ικανότητα να ανακτούν μακροπρόθεσμες εξαρτήσεις και μάλιστα δεν κουράζεται στην εκπαίδευση. Αντίθετα, η απομνημόνευση της γνώσης για μεγάλες χρονικές περιόδους είναι βασικά η προεπιλεγμένη συμπεριφορά τους.[26] Το LSTM εισάγει πύλες για να ελέγχει τι να θυμάται και τι να ξεχνάει πριν την ενημέρωση της κρυφής κατάστασης.



Εικόνα 25: LTSM Networks

Οι νευρώνες LSTM, σε αντίθεση με τους τυπικούς νευρώνες, περιέχουν έναν κλάδο που επιτρέπει στις πληροφορίες να ταξιδεύουν και παρακάμπτουν τη μακρά επεξεργασία του παρόντος κυττάρου, επιτρέποντας την αποθήκευση των αναμνήσεων για εκτεταμένες χρονικές περιόδους. Το πρόβλημα της

κλίσης εξαφάνισης έχει βελτιωθεί, αλλά όχι δραματικά. Για παράδειγμα, λειτουργεί καλά για τις πρώτες 100 λέξεις, αλλά στη συνέχεια χάνει την ροή μετά από περίπου 1.000 λέξεις. Επιπλέον, δέχονται εισόδους διαδοχικά, μία προς μία, γεγονός που περιορίζει τον αποτελεσματικό τρόπο χρήσης των GPU, οι οποίες είναι κατασκευασμένες για παράλληλη επεξεργασία. Όμως, όπως προαναφέρθηκε, ως RNN, χρειάζεται πολύς χρόνος για την εκπαίδευση.[26]

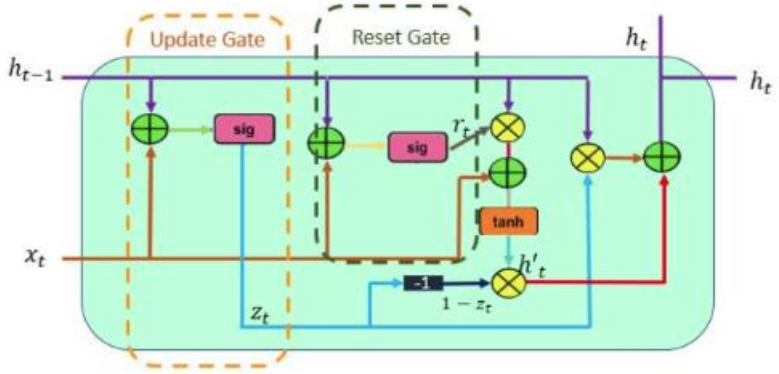
Το πρόβλημα με τη κλίση που εξαφανίζεται λύθηκε με το Attention. Για παράδειγμα, όταν ένας άνθρωπος θέλει να ενημερωθεί για έναν συγκεκριμένο όρο, δεν θα ανατρέξει σε ολόκληρο βιβλίο αλλά θα ανατρέξει κατευθείαν στη σελίδα που χρησιμοποιείται ο όρος. Έτσι, λειτουργεί και το μοντέλο. Εστιάζει στην υψηλή ανάλυση σε ορισμένα μέρη των εισόδων ενώ η υπόλοιπη είσοδος είναι σε χαμηλή ανάλυση.

Εικόνα 26: Δουλεύοντας στο Attention

3.4.3 GRU

Το GRU (GatedRecurrentUnit) είναι βελτιωμένες εκδόσεις τυπικών επαναλαμβανόμενων νευρωνικών δικτύων και στοχεύει στην επίλυση του προβλήματος της εξαφάνισης της κλίσης που συνοδεύεται από ένα τυπικό επαναλαμβανόμενο νευρωνικό δίκτυο. Το GRU μπορεί να θεωρηθεί ως παραλλαγή του LSTM επειδή και τα δύο έχουν σχεδιαστεί παρόμοια και, σε ορισμένες περιπτώσεις, παράγουν εξίσου εξαιρετικά αποτελέσματα. Για να λύσει το πρόβλημα της κλίσης εξαφάνισης ενός τυπικού RNN, η GRU χρησιμοποιεί, τη λεγόμενη, πύλη ενημέρωσης και πύλη επαναφοράς. Στην ουσία, αυτά τα δύο διανύσματα καθορίζουν ποια δεδομένα πρέπει να σταλούν στην έξοδο. Το ιδιαίτερο με αυτά είναι ότι μπορούν να εκπαιδευτούν να διατηρούν πληροφορίες από παλιά, χωρίς να τις ξεχάσουν ή να αφαιρούν πληροφορίες που είναι άσχετες με την πρόβλεψη.[31]Η ροή εργασίας της Gated Recurrent Unit είναι ίδια με το RNN, αλλά η διαφορά έγκειται στη λειτουργία και τις πύλες που σχετίζονται με κάθε μονάδα

GRU. Για να λύσει το πρόβλημα που αντιμετωπίζει το τυπικό RNN, η GRU ενσωματώνει τους δύο μηχανισμούς λειτουργίας πύλης που ονομάζονται πύλη ενημέρωσης και πύλη επαναφοράς.[32]



Εικόνα 27: GRU αρχιτεκτονική

Το GRU όπως προαναφέρθηκε χρησιμοποιεί μια πύλη ενημέρωσης και μια πύλη επαναφοράς. Η πύλη ενημέρωσης είναι υπεύθυνη για τον προσδιορισμό του όγκου των προηγούμενων πληροφοριών που πρέπει να περάσουν στην επόμενη κατάσταση. Αυτό είναι πραγματικά ισχυρό επειδή το μοντέλο μπορεί να αποφασίσει να αντιγράψει όλες τις πληροφορίες από το παρελθόν και να εξαλείψει τον κίνδυνο εξαφάνισης της κλίσης. Η πύλη επαναφοράς χρησιμοποιείται από το μοντέλο για να αποφασίσει πόσες από τις προηγούμενες πληροφορίες χρειάζονται για να αγνοηθούν. Εν ολίγοις, αποφασίζει εάν η προηγούμενη κατάσταση του κελιού είναι σημαντική ή όχι. Αρχικά, η πύλη επαναφοράς τίθεται σε λειτουργία και αποθηκεύει σχετικές πληροφορίες από το προηγούμενο χρονικό βήμα σε νέο περιεχόμενο μνήμης. Στη συνέχεια πολλαπλασιάζει το διάνυσμα εισόδου και την κρυφή κατάσταση με τα βάρη τους και μετέπειτα υπολογίζει τον πολλαπλασιασμό βάσει στοιχείων μεταξύ της πύλης επαναφοράς και του πολλαπλού προηγουμένως κρυφής κατάστασης. Αφού γίνουν τα παραπάνω βήματα, εφαρμόζεται η συνάρτηση μη γραμμικής ενεργοποίησης και δημιουργείται η επόμενη ακολουθία.[32]

Πλέον γίνεται εμφανές πως ο αλγόριθμος GRU είναι όμοιος με τον LSTM, όμως υπάρχουν διαφορές που δείχνουν έντονα τη διαφορά τους. Αρχικά, το GRU έχει δύο πύλες ενώ το LSTM έχει τρεις πύλες. Μια ακόμη διαφορά θεωρείται ότι το GRU δεν διαθέτει εσωτερική μνήμη και δεν έχει πύλη εξόδου που υπάρχει στο LSTM. Στο LSTM η πύλη εισόδου και η πύλη στόχος συνδέονται με μια πύλη ενημέρωσης και μάλιστα στην πύλη επαναφοράς GRU εφαρμόζεται απευθείας στην προηγούμενη κρυφή κατάσταση. Τέλος, στο LSTM την ευθύνη της πύλης επαναφοράς αναλαμβάνουν οι δύο πύλες, δηλαδή η είσοδος και ο στόχος.

Κεφάλαιο 4ο: Ενισχυτική Μάθηση

4.1 Εισαγωγή

Η ενισχυτική μάθηση είναι ένας τομέας της Μηχανικής Μάθησης. Πρόκειται για τη λήψη κατάλληλων μέτρων για τη μεγιστοποίηση της ανταμοιβής σε μια συγκεκριμένη κατάσταση. Χρησιμοποιείται από διάφορα λογισμικά και μηχανήματα για να βρει την καλύτερη δυνατή συμπεριφορά ή διαδρομή που πρέπει να ακολουθήσει σε μια συγκεκριμένη κατάσταση. Η ενισχυτική μάθηση διαφέρει από την εποπτευόμενη μάθηση με τρόπο που στην εποπτευόμενη μάθηση τα δεδομένα εκπαίδευσης έχουν το κλειδί απάντησης, έτσι το μοντέλο εκπαιδεύεται με τη σωστή απάντηση, ενώ στην ενισχυτική μάθηση, δεν υπάρχει απάντηση, αλλά ο ενισχυτικός παράγοντας αποφασίζει τι να κάνει εκτελέσει τη δεδομένη εργασία.[35]

Η ενισχυτική μάθηση χρησιμοποιείται σε πολλούς τομείς σήμερα. Χρησιμοποείται πιο πολύ στον κλάδο της ρομποτικής για βιομηχανικούς αυτοματισμούς, όπου τα ρομπότ που χρησιμοποιούν αυτό το είδος μηχανικής μάθησης μπορεί να είναι σε θέση να μάθουν δεξιότητες που ένας ανθρώπινος εκπαιδευτής δεν μπορεί να εκπαιδεύσει τόσο καλά τις δεξιότητες σε μια νέα εργασία ή δεν μπορεί να επιτύχει στη βελτιστοποίηση, ακόμη και αν δεν υπάρχει αναλυτική διατύπωση.Ακόμη, το RLμπορεί να χρησιμοποιηθεί σε μεγάλα περιβάλλοντα κυρίως όταν ένα μοντέλο του περιβάλλοντος είναι γνωστό, αλλά δεν υπάρχει αναλυτική λύση, όπου δίνεται μόνο ένα μοντέλο προσομοίωσης του περιβάλλοντος και ο μόνος τρόπος συλλογής πληροφοριών για το περιβάλλον είναι η αλληλεπίδραση με αυτό. Τέλος, χρησιμοποιείταιστη μηχανική εκμάθηση, στην επεξεργασία δεδομένων, στην δημιουργία εκπαιδευτικών συστημάτων που παρέχουν προσαρμοσμένες οδηγίες και υλικά σύμφωνα με τις απαιτήσεις των μαθητών, στην επιχειρησιακή έρευνα, στη θεωρία πληροφοριών, στη θεωρία παιγνίων, στη θεωρία ελέγχου, στη βελτιστοποίηση με βάση την προσομοίωση, στα πολυπρακτορικά συστήματα, στη νοημοσύνη σμήνους, στη στατιστική,στους γενετικούς αλγόριθμους και στα παιχνίδια.[34][36]

Επιπλέον, αν και είναι υψηλή σε δυνατότητες, μπορεί να είναι δύσκολο να αναπτυχθεί και παραμένει περιορισμένη στην εφαρμογή της. Ένα από τα εμπόδια για την ανάπτυξη αυτού του τύπου μηχανικής μάθησης είναι η εξάρτησή του από την εξερεύνηση του περιβάλλοντος. Για παράδειγμα, ένα ρομπότ που βασίζεται στην ενισχυτική μάθηση, και έχει τριγύρω του ένα δύσκολο περιβάλλον, σίγουρα θα αναζητούσε νέες καταστάσεις και θα υιοθετούσε νέες συμπεριφορές μέχρι να μάθει πως πρέπει να είναι η συμπεριφορά του σε όλο το τριγύρω περιβάλλον του. Ωστόσο, δεδομένου του πόσο γρήγορααλλάζει το περιβάλλον στον πραγματικό κόσμο, είναι δύσκολο να υπάρχουν με συνέπεια τις σωστές αποφάσεις.

Τα χαρακτηριστικά της ενισχυτικής μάθησης είναι η είσοδο, η έξοδος και η εκπαίδευση. Ειδικότερα, η είσοδος αφορά την αρχική κατάσταση από την οποία ξεκινάει το μοντέλο, όπως φαίνεται και στην παραπάνω εικόνα η θέση του ρομπότ. Όσον αφορά την έξοδο μπορεί να μην είναι πάντα μόνο μια αλλά να υπάρχουν αρκετές εξόδου. Επίσης, η εκπαίδευση βασίζεται στην είσοδο,επειδή το μοντέλο θα επιστρέψει μια κατάσταση και ο χρήστης θα αποφασίσει να επιβραβεύσει ή να τιμωρήσει το μοντέλο με βάση την έξοδο του.Έτσι, το μοντέλο θα συνεχίσει να εκπαιδεύεται.[36]

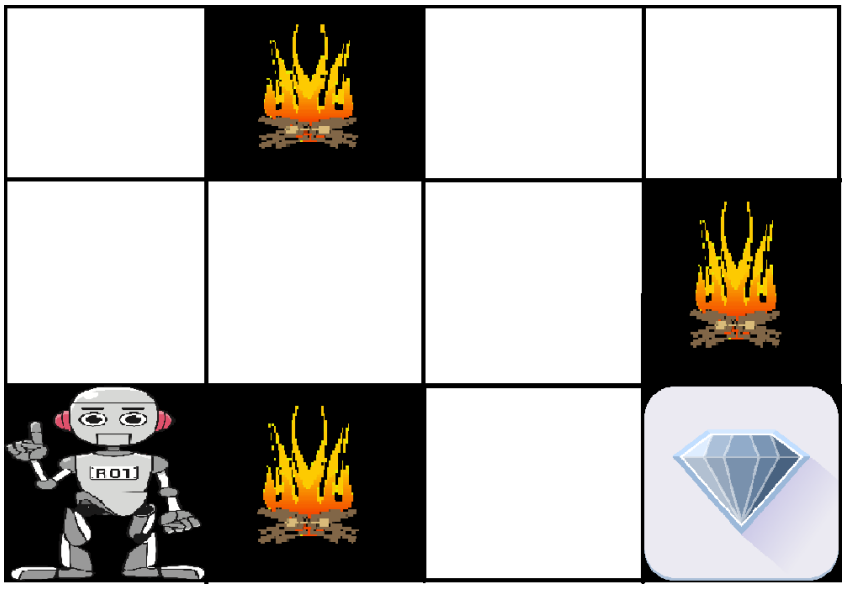
Η ενισχυτική μάθηση διαχωρίζεται σε δύο τύπους, τον θετικό και τον αρνητικό. Συγκεκριμένα, η θετική ενισχυτική ορίζεται ως όταν ένα γεγονός, συμβαίνει λόγω μιας συγκεκριμένης συμπεριφοράς, αυξάνει τη δύναμη και τη συχνότητα της συμπεριφοράς. Με άλλα λόγια, έχει θετική επίδραση στη συμπεριφορά. Το πλεονέκτημα αυτού του τύπου, αφορά την μεγιστοποίηση της απόδοσης, την διατήρηση της αλλαγής γιαμεγάλο χρονικό διάστημα και τηνυπερβολική ενίσχυση, η οποία μπορεί να οδηγήσει σε υπερφόρτωση καταστάσεων που μπορεί να μειώσει τα αποτελέσματα. Ωστόσο, η αρνητική ενίσχυση

ορίζεται ως η ενίσχυση της συμπεριφοράς επειδή μια αρνητική κατάσταση διακόπτεται ή αποφεύγεται. Αυτό επιφέρει ως πλεονεκτήματα, την αύξηση της συμπεριφοράς, την παροχή αντικρίσματος στο ελάχιστο επίπεδο απόδοσης και την παροχή της εκπλήρωσης του ελάχιστου προτύπου συμπεριφοράς. [36]

Η ενισχυτική μάθηση, όπως προαναφέρθηκε, είναι μια μέθοδος εκπαίδευσης μηχανικής μάθησης που βασίζεται στην επιβράβευση επιθυμητών συμπεριφορών ή/και στην τιμωρία των ανεπιθύμητων. Ένας πράκτορας ενίσχυσης μάθησης μπορεί συνήθως να αισθανθεί και να κατανοήσει το περιβάλλον του, να ενεργήσει και να μάθει κάνοντας λάθη. Στην ενισχυτική μάθηση, οι προγραμματιστές επινοούν μια μέθοδο επιβράβευσης, όταν η συμπεριφορά είναι επιθυμητή, και μια μέθοδο τιμωρίας όταν η συμπεριφορά είναι αρνητική. Αυτοί οι μακροπρόθεσμοι στόχοι βοηθούν στην αποτροπή του πράκτορα από το να καθυστερήσει σε μικρότερους στόχους. Ο πράκτορας, με τον καιρό, τελικά μαθαίνει να αποφεύγει τις κακές συμπεριφορές και να ψάχνει για τις καλές. Η τεχνητή νοημοσύνη (AI) έχει υιοθετήσει αυτή τη στρατηγική μάθηση ως τεχνική για τον έλεγχο της μη εποπτευόμενης μηχανικής μάθησης χρησιμοποιώντας κίνητρα και τιμωρίες. Αν και η ενισχυτική μάθηση έχει προσελκύσει μεγάλη προσοχή στον τομέα της τεχνητής νοημοσύνης, εξακολουθούν να υπάρχουν αρκετοί περιορισμοί στην αποδοχή και τη χρήση της στον πραγματικό κόσμο. [34]

Εφόσον υπάρχει μια ξεκάθαρη και σαφής ανταμοιβή, η ενισχυτική μάθηση μπορεί να εφαρμοστεί σε κάθε κατάσταση.Οι αλγόριθμοι ενισχυτικής μάθησης στη διαχείριση πόρων της επιχείρησης (ERM) μπορεί να κατανέμουν σπάνιους πόρους σε διάφορες δραστηριότητες, εφόσον υπάρχει ένας ευρύς στόχος που επιδιώκεται. Η εξοικονόμηση χρόνου ή η διατήρηση πόρων θα ήταν ένας στόχος σε αυτήν την κατάσταση.

Για παράδειγμα, έστω ότι υπάρχει ένας πράκτορας και μια ανταμοιβή με πολλά εμπόδια και ο πράκτορας υποτίθεται ότι πρέπει να βρει την καλύτερη και πιο σύντομη διαδρομή για να φτάσει στην ανταμοιβή (εικόνα 29). Η παρακάτω εικόνα αντιπροσωπεύει το περιβάλλον του παιχνιδιού, όπου παρατηρείται πως η φωτιά είναι τα εμπόδια του ρομπότ και αν η διαδρομή του πέσει πάνω σε ρομπότ, θα χάσει. Στόχος του ρομπότ είναι να φτάσει όσο πιο γρήγορα γίνεται στο διαμάντι, χωρίς να πέσει μέσα σε φωτιά. Έτσι, το ρομπότ μαθαίνει δοκιμάζοντας όλα τα πιθανά μονοπάτια και στη συνέχεια επιλέγοντας το μονοπάτι που του δίνει την ανταμοιβή με τα λιγότερα εμπόδια. Κάθε σωστό βήμα θα δώσει στο ρομπότ μια ανταμοιβή και κάθε λάθος βήμα θα αφαιρέσει την ανταμοιβή του ρομπότ. Η συνολική ανταμοιβή θα υπολογιστεί όταν φτάσει στην τελική ανταμοιβή που είναι το διαμάντι. [36]



Εικόνα 28: Παράδειγμα κατανόησης ενισχυτικής μάθησης

Επίσης, αντί να αναφέρεται σε έναν μόνο αλγόριθμο, ο κλάδος της ενισχυτικής μάθησης περιλαμβάνει έναν αριθμό αλγορίθμων που χρησιμοποιούν διαφορετικές μεθοδολογίες. Οι προσεγγίσεις τους για την πλοήγηση στο περιβάλλον τους ευθύνονται για την πλειοψηφία των διαφορών. Ειδικότερα, ένας αλγόριθμος που χρησιμοποιεί είναι ο λεγόμενος Κράτος-δράση-ανταμοιβή-κράτος-δράση,State-action-reward-state-action , (SARSA), ο οποίος ξεκινά δίνοντας στον πράκτορα αυτό που είναι γνωστό ως πολιτική. Η πολιτική είναι ουσιαστικά μια πιθανότητα που της λέει τις πιθανότητες ορισμένων ενεργειών που καταλήγουν σε ανταμοιβές ή ευεργετικές καταστάσεις. Ένας ακόμμης αλγόρθμος που χησιμοποείται είναι ο Q-learning, όπου υιοθετεί μια διαφορετική στρατηγική. Σε αυτόν τον αλγόριθμο ο πράκτορας δεν λαμβάνει καμία πολιτική, που σημαίνει ότι η εξερεύνηση του περιβάλλοντός του είναι περισσότερο αυτοκατευθυνόμενη. Τέλος, ο τελευταίος αλγόριθμος που χρησιμοποείται είναι ο Deep Q-Networks, όπου συνδυάζει προσεγγίσεις από την ενισχυτική μάθηση με νευρωνικά δίκτυα. Συγκεκριμένα, χρησιμοποιεί ο αυτοκατευθυνόμενο περιβάλλον εξερεύνησης της ενισχυτικής μάθησης και οι μελλοντικές ενέργειες βασίζονται σε ένα τυχαίο δείγμα προηγούμενων ωφέλιμων ενεργειών που μαθαίνονται από το νευρωνικό δίκτυο.

Η μηχανική μάθησηδιαθέτει τέσσερις κατηγορίες μάθησης, οι οποίοι είναι η επίβλεψη χωρίς μάθηση (Supervised learning), η εκμάθηση χωρίς επίβλεψη (Unsupervised learning), η ημιεποπτευόμενη μάθηση (Semisupervised learning) και τέλος η ενισχυτική μάθηση (Reinforcement learning). Συγκεκριμένα, Στην επίβλεψη μάθησης, οι αλγόριθμοι εκπαιδεύονταισε ένα σύνολο δεδομένωνμε ετικέτα και οι αλγόριθμοι μάθησης με επίβλεψη μπορούν να μάθουν μόνο χαρακτηριστικά που καθορίζονται στο σύνολο δεδομένων.Συνήθεις εφαρμογές της εποπτευόμενης μάθησης είναι τα μοντέλα αναγνώρισης εικόνων, τα οποία λαμβάνουν ένα σύνολο εικόνων με ετικέτα και μαθαίνουν να διακρίνουν κοινά χαρακτηριστικά προκαθορισμένων μορφών. Στον αλγόριθμο εκμάθησης χωρίς επίβλεψη, οι προγραμματιστές λύνουν τους αλγόριθμους σε δεδομένα χωρίς ετικέτα, όπου ο αλγόριθμος μαθαίνει καταγράφοντας τις δικές του παρατηρήσεις σχετικά με τα χαρακτηριστικά των δεδομένων χωρίς να έχει οδηγίες για το τι πρέπει να αναζητήσει. Επίσης, όσον αφορά την Ημιεποπτευόμενη μάθησηακολουθεί μια προσέγγιση μεσαίου επιπέδου, δηλαδή οι προγραμματιστές εισάγουν ένα σχετικά μικρό σύνολο δεδομένων εκπαίδευσης με ετικέτα, καθώς και ένα μεγαλύτερο σύνολο δεδομένων χωρίς

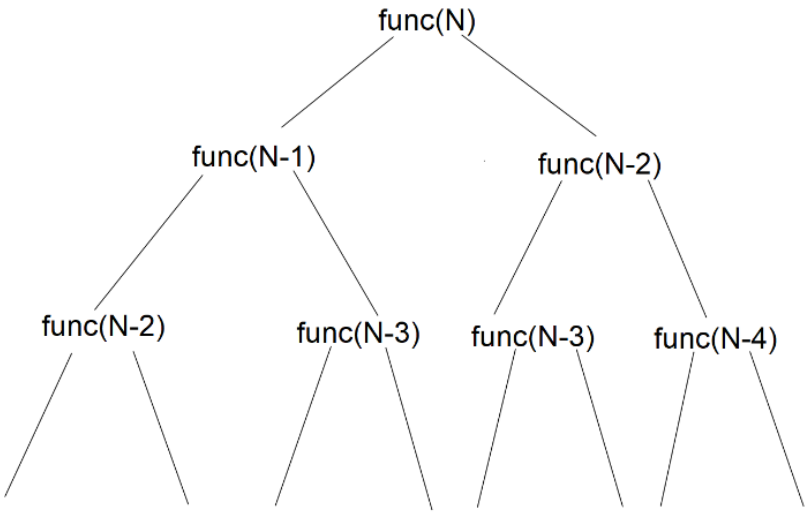
ετικέτα.. Έπειτα, ο αλγόριθμοςαλείται να εφαρμόσει τη γνώση που έχει αποκτήσει από τα δεδομένα με ετικέτα στα δεδομένα χωρίς ετικέτα και να εξάγει συμπεράσματα από το σύνολο ως σύνολο. Τέλος, η ενισχυτική μάθηση έχει μια διαφορετική προσέγγιση, διότι τοποθετεί έναν πράκτορα σε μια κατάσταση με ορισμένα όρια που διαχωρίζουν τη χρήσιμη συμπεριφορά από τη μη ωφέλιμη δράση καθώς και έναν μεγάλο στόχο που πρέπει να επιτύχει.Οιαλγόριθμοιτης ενισχυτικής μάθησης πρέπει να δοθούν σαφώς με καθορισμένους στόχους, καθορισμένες ανταμοιβές και ποινές και αυτό είναι παρόμοιο με την λειτουργία της εποπτευόμενης μάθησης από πολλές απόψεις. Ως αποτέλεσμα, απαιτείται πιο σαφής προγραμματισμός από ό,τι στην εποπτευόμενη μάθηση. Ο αλγόριθμος, ωστόσο, λειτουργεί ανεξάρτητα μετά τον καθορισμό αυτών των παραμέτρων, καθιστώντας τον πολύ πιο αυτοκατευθυνόμενο από τους εποπτευόμενους αλγόριθμους εκμάθησης. Εξαιτίας αυτού, η ενισχυτική μάθηση αναφέρεται περιστασιακά ως ένα υποσύνολο της ημιεποπτευόμενης μάθησης, αν και στην πραγματικότητα, αναγνωρίζεται συχνότερα ως ξεχωριστό υποσύνολο της μηχανικής μάθησης.[34] Δηλαδή, η ενισχυτική μάθηση διαφέρει από την εποπτευόμενη μάθηση με τρόπο που στην εποπτευόμενη μάθηση τα δεδομένα εκπαίδευσης έχουν το κλειδί απάντησης, έτσι το μοντέλο εκπαιδεύεται με τη σωστή απάντηση, ενώ στην ενισχυτική μάθηση, δεν υπάρχει απάντηση, αλλά ο ενισχυτικός παράγοντας αποφασίζει τι να κάνει εκτελέσει τη δεδομένη εργασία.[35]

4.2 Δυναμικός Προγραμματισμός

Ο δυναμικός προγραμματισμός(dynamic programming) χρησιμεύει ως χρήσιμο σημείο έναρξης για την κατανόηση της ενισχυτικής μάθησης.[37] Ειδικότερα, αποτελεί μια μια μαθηματική προσέγγιση βελτιστοποίησης που χρησιμοποιείται συνήθως για τον αυτοσχεδιασμό αναδρομικών αλγορίθμων. Στην ουσία, ο Δυναμικός Προγραμματισμός χρησιμοποιείται κυρίως όταν χρειάζονται λύσεις στα υποπροβλήματακαι περιλαμβάνει την απλοποίηση ενός μεγάλου προβλήματος σε μικρότερα υποπροβλήματα.Για να επιλυθεί ένα ζήτημα χρησιμοποιώντας δυναμικό προγραμματισμό, πρέπει να έχει δύο χαρακτηριστικά, τα Επικαλυπτόμενα υποπροβλήματα (Overlapping Subproblems) και τη Βέλτιστη Υποδομή (Optimal Substructure).[38]

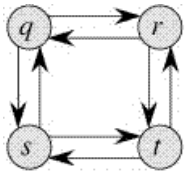
Ο Δυναμικός Προγραμματισμός δεν είναι χρήσιμος όταν δεν υπάρχουν κοινά (επικαλυπτόμενα) υποπροβλήματα γιατί δεν έχει νόημα να αποθηκευτούν οι λύσεις αν δεν χρειάζονται ξανά. Για παράδειγμα, η δυαδική αναζήτηση δεν έχει κοινά υποπροβλήματα. Όμως, με τη χρήση ενός αναδρομικού προγράμματος για τους αριθμούς Fibonacci, υπάρχουν πολλά υποπροβλήματα που λύνονται ξανά και ξανά. Για τα Επικαλυπτόμενα υποπροβλήματα υπάρχουν δύο διαφορετικοί τρόποι αποθήκευσης των τιμών ώστε αυτές οι τιμές να μπορούν να επαναχρησιμοποιηθούν απομνημόνευση (από πάνω προς τα κάτω) και με τη χρήση του πίνακα (από κάτω προς τα πάνω). Όσον αφορά την απομνημόνευση, το απομνημονευμένο πρόγραμμα για ένα πρόβλημα είναι παρόμοιο με την αναδρομική έκδοση με μια μικρή τροποποίηση που εξετάζει έναν πίνακα αναζήτησης πριν υπολογίσει λύσεις. Στην αρχή, γίνεται αρχικοποίηση του πίνακα με αρχικές τιμές ως NIL. Στη συνέχεια, κάθε φορά που χρειάζεται η λύση σε ένα υποπρόβλημα, εξετάζεται πρώτα ο πίνακας αναζήτησης.Αν υπάρχει η προυπολογισμένη τιμή, τότε επιστρέφεται αυτή η τιμή, διαφορετικά, υπολογίζεται η τιμή και μετέπειτα προστίθεται το αποτέλεσμα στον πίνακα αναζήτησης, ώστε να μπορεί να χρησιμοποιηθεί ξανά αργότερα. Επιπλέον, με τη χρήση του πίνακα, το πρόγραμμα με πίνακα για ένα δεδομένο πρόβλημα

δημιουργεί έναν πίνακα με τρόπο από κάτω προς τα πάνω και επιστρέφει την τελευταία καταχώρηση από τον πίνακα.[39]



Εικόνα 29: Overlapping Subproblems

Επιπροσθέτως, η Βέλτιστη υποδομή αφορά ένα δεδομένο πρόβλημα που έχει την ιδιότητα βέλτιστης υποδομής, εάν η βέλτιστη λύση του δεδομένου προβλήματος μπορεί να επιτευχθεί χρησιμοποιώντας βέλτιστες λύσεις των υποπροβλημάτων του.Για παράδειγμα, το πρόβλημα της συντομότερης διαδρομής έχει την ακόλουθη ιδιότητα βέλτιστης υποδομής:Εάν ένας κόμβος x βρίσκεται στη συντομότερη διαδρομή από έναν κόμβο πηγής u προς τον κόμβο προορισμού v, τότε η συντομότερη διαδρομή από το u στο v είναι ένας συνδυασμός της συντομότερης διαδρομής από το u στο x και του συντομότερου μονοπατιού από το x στο v. Το τυπικό ζεύγος όλων Ο αλγόριθμος συντομότερης διαδρομής, όπως ο αλγόριθμος Floyd–Warshall και Single Source Ο αλγόριθμος συντομότερης διαδρομής για ακμές αρνητικού βάρους, όπως το Bellman–Ford, είναι τυπικά παραδείγματα Δυναμικού Προγραμματισμού.Από την άλλη πλευρά, το πρόβλημα Longest Path δεν έχει την ιδιότητα Optimal Substructure. Εδώ με τον όρο Longest Path, εννοούμε τη μεγαλύτερη απλή διαδρομή (διαδρομή χωρίς κύκλο) μεταξύ δύο κόμβων. Για παράδειγμα, το ακόλουθο μη σταθμισμένο γράφημα που δίνεται παρακάτω, έχει δύο μακρύτερα μονοπάτια από το q στο t: q→r→t και q→s→t. Σε αντίθεση με τα συντομότερα μονοπάτια, αυτά τα μεγαλύτερα μονοπάτια δεν έχουν τη βέλτιστη ιδιότητα υποδομής. Για παράδειγμα, το μεγαλύτερο μονοπάτι q→r→t δεν είναι συνδυασμός της μεγαλύτερης διαδρομής από το q στο r και του μεγαλύτερου μονοπατιού από το r στο t, επειδή το μεγαλύτερο μονοπάτι από το q στο r είναι q→s→t→r και το Το μεγαλύτερο μονοπάτι από το r στο t είναι το r→q→s→t. [39]



Εικόνα 30: Παράδειγμα βέλτιστης και χρονοβόρας διαδρομής

Γενικά, για να προσεγγιστεί ένα πρόβλημα θα πρέπει να γίνουν δύο βήματα, μια πρόβλεψη και ένας έλεγχος. Όσον αφορά την πρόβλεψη, θα πρέπει να αξιολογηθεί η πολιτική, δηλαδή να αξιολογηθεί η συνάρτηση αξίας για όλες τις καταστάσεις. Αναφέροντας τον όρο έλεγχος, εννοούμε πως πρέπει να χρησιμοποιηθεί η συνάρτηση υπολογισμένης τιμής για να βελτιωθεί η πολιτική.Χρησιμοποιώντας αυτά τα δύο βήματα επαναληπτικά , βελτιώνεται η πολιτική. [38]

4.3 Markov Decision Process

Μια διαδικασία απόφασης Markov (MDP) είναι μια διαδικασία στοχαστικού ελέγχου διακριτού χρόνου. Το MDP είναι χρήσιμο για τη μελέτη προβλημάτων βελτιστοποίησης που επιλύονται μέσω δυναμικού προγραμματισμού. Το MDP ήταν γνωστό τουλάχιστον από τη δεκαετία του 1950 και ένα βασικό μέρος της έρευνας για τις διαδικασίες λήψης αποφάσεων Markov προέκυψε από το βιβλίο του Ronald Howard του 1960, Dynamic Programming and Markov Processes. Χρησιμοποιείται σε πολλούς κλάδους, συμπεριλαμβανομένης της ρομποτικής, του αυτόματου ελέγχου, της οικονομίας και της κατασκευής. Το όνομα του MDP προέρχεται από τον Ρώσο μαθηματικό Andrey Markov καθώς αποτελούν προέκταση των αλυσίδων Markov.[44]

Το MDP περιλαμβάνει έναν πράκτορα που βρίσκεται μέσα σε περιβάλλον και κάθε φορά αυτός βρίσκεται σε μια κατάσταση. Συγκεκριμένα, ο πράκτορας αποτελεί μια οντότητα την οποία εκπαιδεύουμε ώστε να λαμβάνει σωστές αποφάσεις. Τις αποφάσεις αυτές, τις παίρνει μέσα σε ένα περιβάλλον στο οποίο αλληλοεπιδρά ο πράκτορας. Ο πράκτορας δεν μπορεί να χειριστεί το περιβάλλον. μπορεί να ελέγξει μόνο τις δικές του ενέργειες. Για παράδειγμα, εάν έχει δημιουργηθεί ένα ρομπότ για να μαγειρεύει, θα βρίσκεται συνεχώς μέσα στην κουζίνα και οι ενέργειες που θα εκτελεί θα είναι μόνο μέσα σε αυτόν τον χώρο. Αξίζει να σημειωθεί πως υπάρχει πάντα μια πολιτική, η οποία καθορίζει τη διαδικασία σκέψης πίσω από την επιλογή μιας ενέργειας. Στην πράξη, είναι μια κατανομή πιθανότητας που αποδίδεται στο σύνολο των ενεργειών. Οι ενέργειες με μεγάλη ανταμοιβή θα έχουν μεγάλη πιθανότητα και το αντίστροφο.[42]

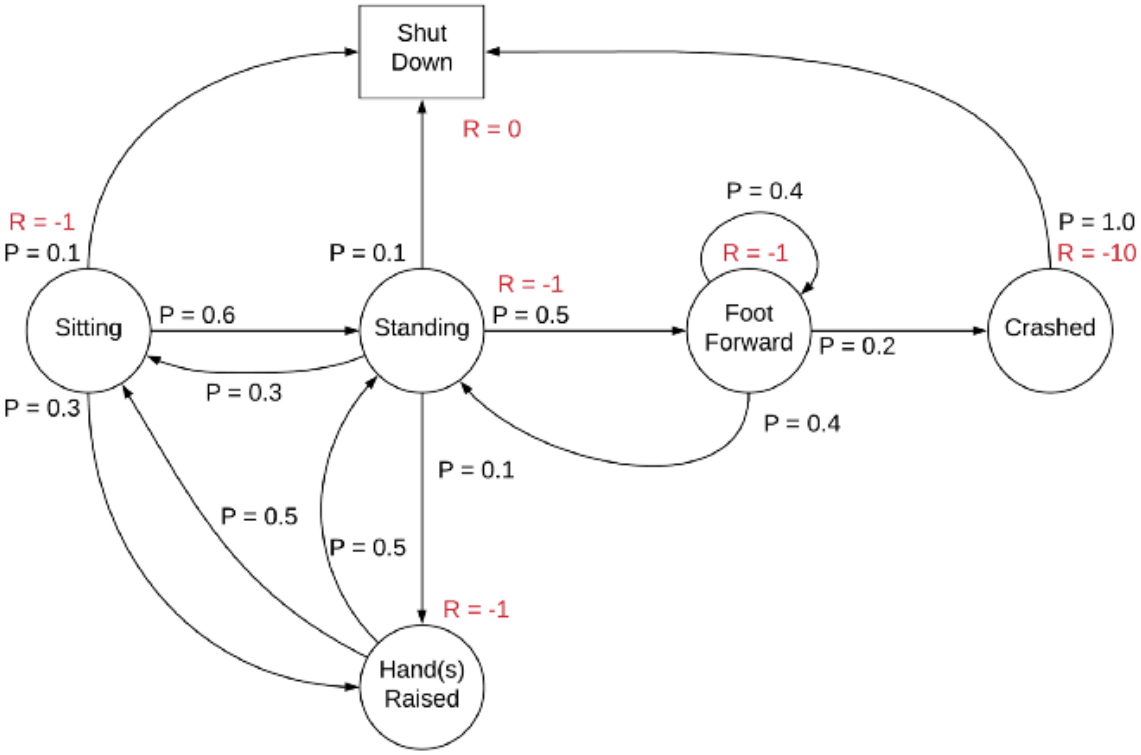
Ένας πράκτορας εξαρτάται μόνο από την άμεση προηγούμενη κατάστασή του (ή το προηγούμενο χρονικό βήμα), δηλαδή την κατάσταση στην οποία βρισκόταν. Ομοίως, η επόμενη κατάστασή του εξαρτάται μόνο από την τρέχουσα κατάστασή του. Για παράδειγμα, Έστω ότι ένα ρομπότ καθόταν και μετέπειτα σηκώθηκε και κατευθύνθηκε προς την πόρτα του σπιτιού. Η τωρινή κατάσταση από την οποία εξαρτάται είναι αυτή που στέκεται τώρα στην πόρτα του σπιτιού και όχι από προηγούμενες ενέργειες.

$$P[s_{t+1}|s_t] = P[s_{t+1}|s_1 \dots, s_t]$$

Μια διεργασία Markov ορίζεται από το (S, P) όπου S είναι οι καταστάσεις και P είναι η πιθανότητα μετάβασης κατάστασης. Αποτελείται από μια ακολουθία τυχαίων καταστάσεων S1, S2, … όπου όλες οι καταστάσεις υπακούουν στην Ιδιότητα Markov.

$$P[ss'] = P[s_{t+1} = s'|s_t = s]$$

Ένα MRP ορίζεται από το (S, P, R, γ), όπου S είναι οι καταστάσεις, P είναι η πιθανότητα μετάβασης κατάστασης, R_s είναι η ανταμοιβή και γ είναι ο συντελεστής έκπτωσης. Η ανταμοιβή κατάστασης R_s είναι η αναμενόμενη ανταμοιβή σε όλες τις πιθανές καταστάσεις στις οποίες μπορεί κανείς να μεταβεί από την κατάσταση s. Αυτή η ανταμοιβή λαμβάνεται επειδή βρίσκεται στην κατάσταση S_t. Επιπλέον, ανταμείβεται μετά την αποχώρηση του πράκτορα από την κατάσταση και, ως εκ τούτου, θεωρείται ως R_(t+1). Για παράδειγμα:



Εικόνα 31: Παράδειγμα Markov

Οι τύποι διαδικασίας μια επιβράβευσης Markov δίνεται από τον παρακάτω τύπο:

$$R_5 = E[R_{t+1}|s_t = s]$$

Ένα MDP ορίζεται από το (S, A, P, R, γ), όπου A είναι το σύνολο των ενεργειών. Είναι ουσιαστικά MRP με ενέργειες. Η εισαγωγή στις ενέργειες προκαλεί μια έννοια ελέγχου της Διαδικασίας Markov, δηλαδή, προηγουμένως, η πιθανότητα μετάβασης της κατάστασης και οι ανταμοιβές κατάστασης ήταν περισσότερο ή λιγότερο στοχαστικές (τυχαίες). Ωστόσο, τώρα οι ανταμοιβές και η επόμενη κατάσταση εξαρτώνται επίσης από την ενέργεια που επιλέγει ο πράκτορας. Βασικά, ο πράκτορας μπορεί πλέον να ελέγξει τη μοίρα του (σε κάποιο βαθμό).

$$\rho_{ss'}^{a} = P[S_{t+1} = s'|S_t = a, A_t = a]$$

$$R_s^a = E[R_{t+1}|S_t = a, A_t = a]$$

Οι ανταμοιβές είναι προσωρινές. Ακόμη και μετά την επιλογή μιας ενέργειας που δίνει μια αξιοπρεπή ανταμοιβή, μπορεί να μας λείπει μια μεγαλύτερη συνολική ανταμοιβή μακροπρόθεσμα. Αυτή η μακροπρόθεσμη συνολική ανταμοιβή είναι η Επιστροφή. Ωστόσο, στην πράξη, θεωρούμε εκπτωτικές Επιστροφές.

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{K=0}^{\infty} \gamma^k R_{t+k+1}$$

Η μεταβλητή γ ∈ [0, 1] στο σχήμα είναι ο συντελεστής έκπτωσης. Η διαίσθηση πίσω από τη χρήση μιας έκπτωσης είναι ότι δεν υπάρχει βεβαιότητα για τις μελλοντικές ανταμοιβές. Δηλαδή, όσο σημαντικό

είναι να ληφθούν οι μελλοντικές ανταμοιβές για την αύξηση της απόδοσης, είναι εξίσου σημαντικό ο περιορισμός της συμβολής ων μελλοντικών ανταμοιβών στην Επιστροφή.

Όπως αναφέρθηκε προηγουμένως, μια πολιτική ορίζει τη σκέψη πίσω από τη λήψη μιας απόφασης (επιλέγοντας μια ενέργεια). Καθορίζει τη συμπεριφορά ενός πράκτορα RL. Τυπικά, μια πολιτική είναι μια πιθανότητα επιλογής μιας ενέργειας a στην κατάσταση s.

$$\Pi(\alpha|5)=P[A_t = \alpha|S_t= s]$$

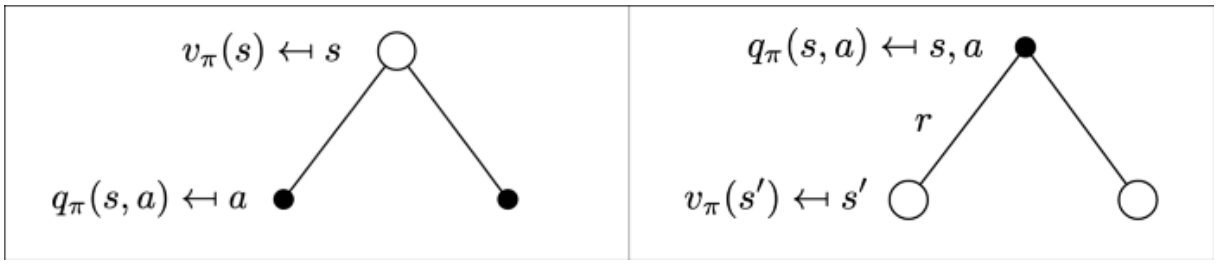
Η εξίσωση Bellman δίνει μια τυπική αναπαράσταση για συναρτήσεις τιμών. Ουσιαστικά διασπά τη συνάρτηση τιμής σε δύο στοιχεία, στην άμεση ανταμοιβή R_(t+1), και στην προεξοφλημένη τιμή της μελλοντικής κατάστασης γ.v(S_(t+1)).

Για παράδειγμα, έστω ότι έχουμε έναν πράκτορα, ο οποίος μεταβαίνει από την τρέχουσα κατάσταση s σε κάποια κατάσταση s, Τώρα, η συνάρτηση τιμής κατάστασης είναι βασικά η αναμενόμενη τιμή των Επιστροφών σε όλα τα s. Τώρα, χρησιμοποιώντας τον ίδιο ορισμό, μπορούμε να αντικαταστήσουμε αναδρομικά την Επιστροφή της επόμενης κατάστασης s με τη συνάρτηση τιμής του s. Αυτό ακριβώς κάνει η εξίσωση Bellman:

$$u(s) = \Sigma[_{R_{t+1}+\gamma v}(s_{t+1})|s_t = s]$$

Έτσι, δεδομένου ότι η προσδοκία είναι διανεμητική, μπορούμε να λύσουμε και για το R_(t+1) και το v(s') ξεχωριστά. Έχουμε ήδη δει ότι η αναμενόμενη τιμή του R_(t+1) έναντι του S_t=s είναι η ανταμοιβή R_s. Και η προσδοκία του v(s') έναντι όλων των s' λαμβάνεται από τον ορισμό της Αναμενόμενης Τιμής.

Δεδομένου ότι γνωρίζουμε τα βασικά της εξίσωσης Bellman τώρα, μπορούμε να μεταβούμε απευθείας στη λύση αυτής της εξίσωσης και να δούμε πώς αυτή διαφέρει από την εξίσωση Bellman για τα MRP:



Εικόνα 32: Διαφορά Bellman – Markov

Αξίζει να σημειωθεί πως οι καταστάσεις χρησιμοποιούνται σε κύκλο και οι ενέργειες και τελείες. Και τα δύο παραπάνω διαγράμματα είναι μια όψη διαφορετικού επιπέδου του ίδιου MDP, με το αριστερό να είναι η προβολή με επίκεντρο το κράτος και το δεξί να είναι η προβολή με επίκεντρο τη δράση. Στο πρώτο σχήμα, ο κύκλος με τη τελεία αντιπροσωπεύει την κατάσταση s που βρίσκεται ο πράκτορας. Εκεί επιλέγει μια ενέργεια σύμφωνα με την πολιτική που υπάρχει. Αυτό το μέρος είναι πλήρως ελεγχόμενο από τον πράκτορα καθώς μπορεί να επιλέξει τη δράση. Αντίθετα, ο ανοιχτός κύκλος αντιπροσωπεύει το περιβάλλον που ενεργεί στον πράκτορα και τον στέλνει σε κατάσταση με βάση την πιθανότητα μετάβασης. Αυτό το μέρος δεν είναι ελεγχόμενο από τον πράκτορα, καθώς δεν μπορεί να ελέγξει πώς ενεργεί το περιβάλλον, απλώς τη δική του συμπεριφορά. Τώρα θα αντιμετωπιστούν δύο μεταβάσεις:

$$U_\pi(S) = \sum_{a \in A} \pi(\alpha|s), q_\pi\ (s, a)$$

Εφόσον έχουμε μετάβαση από κατάσταση σε δράση, λαμβάνουμε την αναμενόμενη τιμή ενέργειας σε όλες τις ενέργειες. Και αυτό ικανοποιεί πλήρως την εξίσωση Bellman καθώς το ίδιο γίνεται και για τη συνάρτηση τιμής δράσης:

$$q\pi(s, a) = R_s^a + \gamma \sum_{s \in s\prime} \rho_{ss\prime}^a U_\Pi(s')$$

Μπορούμε να αντικαταστήσουμε αυτήν την εξίσωση στη συνάρτηση τιμής κατάστασης για να λάβουμε την τιμή από την άποψη των αναδρομικών συναρτήσεων τιμής κατάστασης (και αντίστροφα) παρόμοια με τα MRP:

$$U_\pi(S) = \sum_{a \in A} \pi(\alpha|s)\ (\ R_s^a + \ \gamma \sum_{s \in s\prime} \rho_{ss\prime}^a U_\Pi(s'))$$

$$q\pi(s, a) = R_s^a + \gamma \sum_{s \in s'} \rho_{ss\prime}^a\ q_\pi(s', a')$$

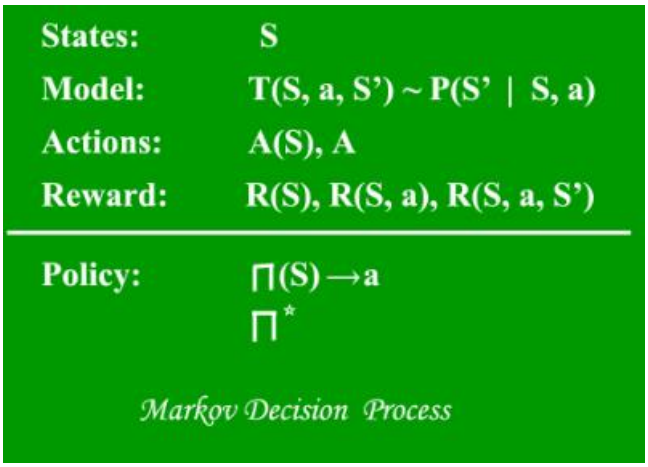
Έστω ότι λήφθηκαν όλες οι καταστάσεις ενός MDP για όλα τα πιθανά πρότυπα ενεργειών που μπορούν να επιλεγούν, δηλαδή όλες τις πολιτικές. Στη συνέχεια, πρέπει να επιλεχτεί η πολιτική με την υψηλότερη αξία για τις πολιτείες και τις ενέργειες. Οι παρακάτω εξισώσεις αντιπροσωπεύουν αυτό ακριβώς το πράγμα.

$$U^*(s)=\max_\Pi v_\pi(S)$$

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

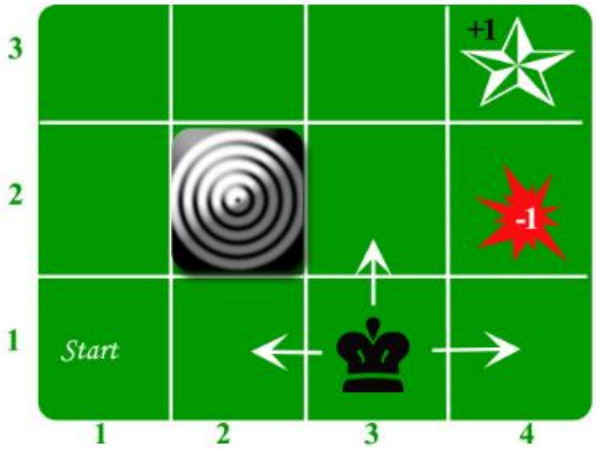
Τώρα απλά εκχωρείται η πιθανότητα 1 για την ενέργεια που έχει τη μέγιστη τιμή για q∗ και 0 για τις υπόλοιπες ενέργειες δηλαδή για όλες τις δεδομένες καταστάσεις. Εφόσον ούτως ή άλλως θα διαλέξουμε την ενέργεια που αποδίδει το μέγιστο q∗, μπορούμε απλά να αντιστοιχίσουμε αυτήν την τιμή ως τη βέλτιστη συνάρτηση τιμής. Ωστόσο, εφόσον ακολουθούμε τη βέλτιστη πολιτική, η συνάρτηση τιμής κατάστασης θα είναι η βέλτιστη.[42]

Ένα MDP (Markov Decision Process) έχει συγκεκριμένα χαρακτηριστικά. Αρχικά, περιλαμβάνει ένα σύνολο πιθανών παγκόσμιων κρατών S, όπου ένα κράτος αποτελεί ένα σύνολο διακριτικών που αντιπροσωπεύουν κάθε κατάσταση στην οποία μπορεί να βρίσκεται ο πράκτορας. Επίσης, περιλαμβάνει ένα σύνολο μοντέλων, το οποίο δίνει το αποτέλεσμα μιας ενέργειας σε μια κατάσταση. Συγκεκριμένα, το T(S, a, S') ορίζει μια μετάβαση T όπου ο πράκτορας βρίσκεται στην κατάσταση S και κάνει μια ενέργεια «a» και οδηγείται στην κατάσταση S» (S και S» μπορεί να είναι τα ίδια). Για στοχαστικές ενέργειες (θορυβώδεις, μη ντετερμινιστικές) ορίζουμε επίσης μια πιθανότητα P(S'|S,a) η οποία αντιπροσωπεύει την πιθανότητα επίτευξης μιας κατάστασης S' εάν η ενέργεια 'a' εκτελείται στην κατάσταση S. Επιπλέον, ένα ακόμη χαρακτηριστικό του MDP αποτελεί το σύνολο πιθανών ενεργειών A, όπου μια ενέργεια A είναι ένα σύνολο από όλες τις πιθανές ενέργειες. Το A(s) ορίζει το σύνολο των ενεργειών που μπορούν να γίνουν στην κατάσταση S. Επιπροσθέτως, ένα ακόμη χαρακτηριστικό αποτελεί η συνάρτηση ανταμοιβής με πραγματική αξία R(s,a), όπου η ανταμοιβή είναι μια συνάρτηση ανταμοιβής με πραγματική αξία. Το R(s) υποδηλώνει την ανταμοιβή για το ότι ο πράκτορας βρίσκεται στην κατάσταση S. Το R(S,a) υποδεικνύει την ανταμοιβή για το ότι ο πράκτορας βρίσκεται ε σε κατάσταση S και κάνει μια ενέργεια «a». Το R(S,a,S) υποδεικνύει την ανταμοιβή για το ότι βρίσκεται σε μια κατάσταση S, και εκτελεί μια ενέργεια «a» και στο τέλος καταλήγει σε κατάσταση S. Τέλος, άλλο ένα χαρακτηριστικό αποτελεί μια πολιτική, η οποία αποτελεί μια λύση στη Διαδικασία Απόφασης Markov.[43]



Εικόνα 33: Markov Decision Process

Όπως προαναφέρθηκε η πολιτική αποτελεί ένα χαρακτηριστικό του MDP. Συγκεκριμένα, μια πολιτική είναι μια αντιστοίχιση από το S στο a. Υποδεικνύει την ενέργεια «a» που πρέπει να γίνει ενώ ο πράκτορας βρίσκεται στην κατάσταση S. Για παράδειγμα, έστω ότι υπάρχει ένας πράκτορας που ζει στο παρακάτω πλέγμα, το οποίο είναι 3·4. Η αρχή όπως φαίνεται παρακάτω είναι από το σημείο (1,1), όπου αναγράφεται η λέξη start. Στόχος του παιχνιδιού είναι ο πράκτορας να περιπλανηθεί στο πλέγμα για να φτάσει τελικά στο αστέρι, σημείο (4,3). Σε κάθε περίπτωση, ο πράκτορας θα πρέπει να αποφεύγει το φωτιά στο σημείο (4,2). Εάν ο πράκτορας φτάσει στη φωτιά, η ανταμοιβή που του δίνεται είναι -1. Όμως, το σημείο (2,2), το οποίο περιλαμβάνει έναν στόχο, λειτουργεί ως τοίχος όπου ο πράκτορας δεν μπορεί να τον διαπεράσει και μάλιστα δεν μπορεί να εισέλθει σε αυτό. Επίσης, έστω ότι οι κινήσεις του πράκτορα είναι μόνο δεξιά, αριστερά, πάνω, κάτω. Τέλος, οι τελικές ανταμοιβές έρχονται στο τέλος (καλές ή κακές) και στόχος είναι να μεγιστοποιηθεί το άθροισμα των ανταμοιβών. [43]



Εικόνα 34: Παράδειγμα Markov

Ως πρώτος στόχος θεωρείται να βρεθεί η συντομότερη ακολουθία που φτάνει από την έναρξη στο διαμάντι. Δεδομένου αυτού, μπορούν να βρεθούν δύο επιλογές, η δεξιά-δεξιά-πάνω-πάνω- δεξιά και η πάνω-πάνω-δεξιά-δεξιά-δεξιά. Έστω ότι ο πράκτορας ακολουθεί τη δεύτερη επιλογή. Αυτή η επιλογή είναι πολύ θορυβώδης. Η αναμενόμενη δραστηριότητα εμφανίζεται με ακρίβεια στο 80% των περιπτώσεων. Το 20% του χρόνου που χρειάζεται ο παράγοντας δράσης τον κάνει να κινείται σε ορθή

γωνία. Για παράδειγμα, εάν ο πράκτορας πηγαίνει πάνω , η πιθανότητα να πάει επάνω είναι 0,8 ενώ η πιθανότητα να πάει αριστερά είναι 0,1 και η πιθανότητα να πάει δεξιά είναι 0,1.

4.4 Policy Iteration

Το Policy Iteration είναι ένας αλγόριθμος της Ενισχυτικής Μάθησης, ο οποίος βοηθά στην εκμάθηση της βέλτιστης πολιτικής που μεγιστοποιεί τη μακροπρόθεσμη ανταμοιβή. Αυτές οι τεχνικές είναι συχνά χρήσιμες, όταν υπάρχουν πολλές επιλογές και κάθε επιλογή έχει τα δικά της οφέλη και κινδύνους.

Όπως είναι φυσικό υπάρχουν και θετικές και αρνητικές ανταμοιβές. Για παράδειγμα, έστω ότι υπάρχει ένα πειρατικό πλοίο, το οποίο είναι σταματημένο σε ένα νησί και πρέπει να φτάσει στην πατρίδα του με ασφάλεια. Μέχρι να φτάσει το πλοίο στη πατρίδα του πρέπει να σταματήσει σε άλλα νησιά για να μαζέψει χρυσό και να γεμίσει με καύσιμα. Όμως, υπάρχουν διάφοροι κίνδυνοι στα νησιά. Άλλωτε μπορεί να κατευθυνθεί βόρεια και να φτάσει σε νησί που έχει χρυσό και μετέπειτα να μετακινηθεί νότια για να φτλασει στην πατρίδα, άλλωτε μπορεί να φτάσει στο τρίγωνο των Βερμούδων, στα βόρεια του χρυσού νησιού, όπου εκεί το πλοίο θα ρουφηχτεί και θα χαθεί για πάντα, άλλωτε μπορεί το νησί να είναι γεμάτο από ασήμι και άλλωτε μπορεί να βρεθεί στο νησί φυλακών στα νότια του ασημένιου νησιού, όπου εκεί το πλήρωμα φυλακίζεται. Η διασκέδαση του παιχνιδιού εμφανίζεται με την προσθήκη ενός καπετανιού, όπου κάθε φορά που ο καπετάνιος δίνει οδηγία στο πλήρωμα του να προσανατολίσουν το πλοίο βόρια, τότε κινείται βόρεια με πιθανότητα 0,8. Ωστόσο, μπορεί να χάσει το σημάδι και να φτάσει νότια με πιθανότητα 0,2. Ομοίως, αν κινηθεί νότια, υπάρχει πιθανότητα 0,8 να πάει νότια και 0,2 πιθανότητα να πάει βόρεια.

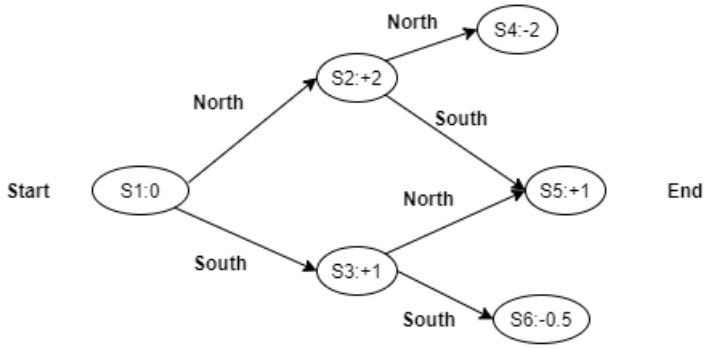
Στη συνέχεια, πρέπει να οριστούν οι κανόνες-ανταμοιβές του παιχνιδιού, για παράδειγμα, εάν το πειρατικό πλοίο φτάσει στην πατρίδα συγκεντρώνει +1 βαθμό, εάν φτάσει σε νησί και υπάρχει χρυσός, θα ανταμοιφθεί +2 βαθμούς, εάν φτάσει σε νησί με ασημένιο χρυσό, θα συγκεντρώσει +1 βαθμό,ακόμη εάν το πλοίο απορροφηθεί στο τρίγωνο των Βερμούδων, τότε το πλοίο συγκεντρώνει -2 βαθμούς και τέλος εάν το πλοίο αιχμαλωτιστεί στο νησί της φυλακής, τότε το πλοίο συγκεντρώνει -0,5 βαθμούς. Στόχος είναι να φτάσει το πειρατικό πλοίο στη πατρίδα του με τη πιο βέλτιση διαδρομή, έχοντας πολλές ανταμοιβές. Έτσι, λειτουργεί αυτός ο αλγόριθμος, ξεκινώντας από την αρχικοποίηση τυχαίας πολιτικής, συνεχίζει μ ετην αξιολόγηση πολιτικής και τέλος τη βελτιώνει.

Όσον αφορά τον όρο πολιτική, εννοούμε μια αντιστοίχιση μιας ενέργειας σε κάθε πιθανή κατάσταση στο σύστημα. Μια βέλτιστη πολιτική είναι εκείνη η πολιτική που μεγιστοποιεί τη μακροπρόθεσμη ανταμοιβή. Για παράδειγμα, μπορεί να υπάρχουν πολλαπλές πολιτικές, δηλαδή πολλαπλές ενέργειες σε κάθε νησί που υπάρχει μόνο μια πολιτική που δίνει την τελική μέγιστη ανταμοιβή. Στόχος είναι να βρεθεί αυτή η βέλτιστη πολιτική.[40]

Τα βήματα αυτού του αλγορίθμου είναι τρία. Ως πρώτο βήμα θεωρείται η αρχικοποίηση μιας τυχαίας πολιτικής όπου στο πρώτο βήμα ξεκινάνε οι ενέργειες τυχαία. Ως δεύτερο βήμα θεωρείται η εξίσωση του Bellman $V_{(s)} = r(s) + \gamma \cdot max\left(\sum_{s',r} p(s',r|s,\pi(s))\right) \cdot v(s')$, όπου λαμβάνεται κάθε ενέργεια για κάθε κατάσταση στην πολιτική και αξιολογείται η συνάρτηση τιμής χρησιμοποιώντας την παραπάνω εξίσωση. Το p είναι η πιθανότητα μετάβασης για παράδειγμα τι πιθανότητα είχε το πειρατικό πλοίο να πάει βόρεια. Για κάθε κατάσταση, λαμβάνεται η καλύτερη ενέργεια από τη συνάρτηση τιμής χρησιμοποιώντας $\pi(s) = arg\,max\, a\sum_{s'r} p\,(s',r|s,a) \cdot V_{(s')}$. Εάν η καλύτερη ενέργεια είναι καλύτερη από την τρέχουσα ενέργεια πολιτικής, τότε αντικαθιστάται από την τρέχουσα ενέργεια με την καλύτερη

ενέργεια. Τα τελευταία δύο βήματα επαναλαμβάνονται μέχρι τη σύγκλιση. Εάν η πολιτική δεν άλλαξε κατά τη διάρκεια μιας επανάληψης, τότε θεωρείται ότι ο αλγόριθμος έχει συγκλίνει.

Επιπλέον, αυτός ο αλγόριθμος περιλαμβάνει ένα διάγραμμα μετάβασης κατάστασης. Ως πρώτο βήμα θα πρέπει να είναι η κατανόηση των καταστάσεων και ενεργειώνγια τη δημιουργία ενός διαγράμματος μετάβασης κατάστασης. Για παράδειγμα, κάθε νησί είναι μια κατάσταση και υπάρχουν δύο ενέργειες, «βόρεια» και «νότια». Με βάση αυτά τα γεγονότα, μπορούμε να δημιουργήσουμε ένα διάγραμμα μετάβασης κατάστασης ως εξής: 6 καταστάσεις, συμπεριλαμβανομένων της κατάστασης εκκίνησης και προορισμού και τέσσερα ενδιάμεσα νησιά στα οποία μπορούν να μεταβούν. Ας χαρακτηριστούν οι καταστάσεις από S1 έως S6 ως εξής:



Εικόνα 35: Παράδειγμα Policy Iterration

Όπου, S1 θεωρείται η κατάσταση εκκίνησης, S2 η Κατάσταση προσγείωσης στο Gold Island, S3 κατάσταση προσγείωσης στο Silver Island, S4 κατάσταση προσγείωσης στο νησί του Τριγώνου των Βερμούδων, S5 κατάσταση προσγείωσης στο νησί προορισμού και S6 η κατάσταση προσγείωσης στο νησί της φυλακής. Δεδομένου ότι το παιχνίδι είναι στοχαστικό, πρέπει να υπολογίσουμε τις πιθανότητες μετάβασης για κάθε ζεύγος κατάστασης/ενέργειας. Με βάση τις πιθανότητες που παρέχονται παραπάνω και το διάγραμμα μετάβασης κατάστασης, υπάρχουν δύο ενέργειες και χρειαζόμαστε έναν πίνακα πιθανοτήτων μετάβασης και για τις δύο αυτές ενέργειες. Σημειώνεται ότι κατά την εφαρμογή της εξίσωσης του Bellman, το T(S, a, S') αναφέρεται στην πιθανότητα μετάβασης από την κατάσταση S στην κατάσταση S μετά την εκτέλεση μιας ενέργειας «a».

$$T[A(North)] \begin{bmatrix} & S1 & S2 & S3 & S4 & S5 & S6 \\ S1 & 0 & 0.8 & 0.2 & 0 & 0 & 0 \\ S2 & 0 & 0 & 0 & 0.8 & 0.2 & 0 \\ S3 & 0 & 0 & 0 & 0 & 0.8 & 0.2 \\ S4 & 0 & 0 & 0 & 0 & 0 & \\ S5 & 0 & 0 & 0 & 0 & 0 & \\ S6 & 0 & 0 & 0 & 0 & 0 & \end{bmatrix}$$

Transition Probability Matrix for Action North

$$T[A(South)] \begin{bmatrix} & S1 & S2 & S3 & S4 & S5 & S6 \\ S1 & 0 & 0.2 & 0.8 & 0 & 0 & 0 \\ S2 & 0 & 0 & 0 & 0.2 & 0.8 & 0 \\ S3 & 0 & 0 & 0 & 0 & 0.2 & 0.8 \\ S4 & 0 & 0 & 0 & 0 & 0 & \\ S5 & 0 & 0 & 0 & 0 & 0 & \\ S6 & 0 & 0 & 0 & 0 & 0 & \end{bmatrix}$$

Transition Probability Matrix for Action South

Εικόνα 36: Transition Probability Matrix

Με αυτές τις πληροφορίες, ας εφαρμόσουμε τον προαναφερθέντα αλγόριθμο βήμα προς βήμα. Έστω ότι ο παράγοντας γ είναι 1. Η αρχικοποίηση της τυχαίας πολιτικής ως κινούμενη βόρεια για όλες τις πολιτείες είναι P = {N, N, N, N, N, N}. Παρατηρώντας το διάγραμμα μετάβασης κατάστασης, βλέπουμε πως οι καταστάσεις S4, S5, S6 δεν υποστηρίζουν ενέργειες σε αυτές τις καταστάσεις, καθώς πρόκειται για τελικές καταστάσεις. Για αυτό θα πρέπει να περιοριστεί η πολιτική ώστε να ισχύει μόνο στις τρεις πρώτες πολιτείες όπου μπορεί να αναλάβει δράση P = {N, N, N}. Έστω ότι η αρχική τιμή V(s) για όλες τις καταστάσεις είναι 0. Έτσι, η εξίσωση Bellman θα μειωθεί σε V(s) = R(s), όπου το R(s) είναι η ανταμοιβή για την είσοδο σε μια κατάσταση. Η αξιολόγηση της πολιτικής για τη πρώτη επανάληψη είναι η εξής:

V[S1] = 0; V[S4] = -2

V[S2] = 2;

V[S3] = 1; V[S6] = -0,5

State	Action	V[S]	Max Action
S1	North	V[S] = 0.8 * 2 + 0.2 * 1 = 1.8	
	South	V[S] = 0.2 * 2 + 0.8 * 1 = 1.2	
			North
S2	North	V[S] = 0.8 * -2 + 0.2 * 1 = -1.4	
	South	V[S] = 0.2 * -2 + 0.8 * 1 = 0.4	South
S3	North	V[S] = 0.8 * 1 + 0.2 * -0.5 = 0.7	North
	South	V[S] = 0.2 * 1 + 0.8 * -0.5 = -0.2	

Εικόνα 37:Βελτίωση της Πολιτικής (1)

Η πολιτική που προκύπτει με βάση τον παραπάνω πίνακα είναι η εξής: P = {N,S,N}

Η αξιολόγηση πολιτικής για τη δεύτερη επανάληψη έχει τις ακόλουθες τιμές:

V[S1] = 3; V[S4] = -2

V[S2] = 1; V[S5] = 1

V[S3] = 1.5; V[S6] = -0.5

Η βελτίωση της πολιτικής για τη δεύτερη επανάληψη είναι P = {S, S, N}.

State	Action	V[S]	Max Action
S1	North	V[S] = 0.8 * 1 + 0.2 * 1.5 = 1.1	
	South	V[S] = 0.2 * 1 + 0.8 * 1.5 = 1.4	
			South
S2	North	V[S] = 0.8 * -2 + 0.2 * 1 = -1.4	
	South	V[S] = 0.2 * -2 + 0.8 * 1 = 0.4	South
S3	North	V[S] = 0.8 * 1 + 0.2 * -0.5 = 0.7	North
	South	V[S] = 0.2 * 1 + 0.8 * -0.5 = -0.2	

Εικόνα 38: Βελτίωση της πολιτικής (2)

Ακόμη, όσον αφορά την πολιτική της τρίτης επανάληψης είναι οι εξής:

State	Action	V[S]	Total value (Reward +Sum of Value for each Action)
S1	North	V[S] = (0.8* 1 + 0.2 * 1.5) = 1.1	
	South	V[S] = (0.2* 1 + 0.8 * 1.5) = 1.4	
			0 + 1.1 + 1.4 = 2.5
S2	North	V[S] = (0.8* -2 + 0.2 * 1) = -1.4	
	South	V[S] = (0.2* -2 + 0.8 * 1) = 0.4	2 -1.4 + 0.4 = 1
S3	North	V[S] = 0.8 * 1 + 0.2 * -0.5 = 0.7	
	South	V[S] = 0.2 * 1 + 0.8 * -0.5 = -0.2	1 + 0.7 -0.2 = 1.5

Εικόνα 39: Βελτίωση της πολιτικής (3)

Οι τιμές για κάθε κατάσταση θα μπορούσαν να συνοψιστούν ως εξής:

$$V[S1] = 2,5;\ V[S4] = -2$$

$$V[S2] = 1;\ V[S5] = 1$$

$$V[S3] = 1,5;\ V[S6] = -0,5$$

Η βελτίωση της τρίτης επανάληψης είναι P = {S, S, N}.

State	Action	V[S]	Max Action
S1	North	V[S] = 0.8 * 1 + 0.2 * 1.5 = 1.1	
	South	V[S] = 0.2 * 1 + 0.8 * 1.5 = 1.4	
			South
S2	North	V[S] = 0.8 * -2 + 0.2 * 1 = -1.4	
	South	V[S] = 0.2 * -2 + 0.8 * 1 = 0.4	South
S3	North	V[S] = 0.8 * 1 + 0.2 * -0.5 = 0.7	North
	South	V[S] = 0.2 * 1 + 0.8 * -0.5 = -0.2	

Εικόνα 40: Βελτίωση της πολιτικής (4)

Παρατηρώντας πως αυτή η πολιτική, με την πολιτική που υπήρξε στη δεύτερη επανάληψη δεν άλλαξε, πράγμα που σημαίνει ότι ο αλγόριθμος έχει συγκλίνει και αυτή είναι η βέλτιστη πολιτική.

Συμπερασματικά, βρέθηκε η βέλτιστη πολιτική ως {South, South, North} εφαρμόζοντας τον αλγόριθμο. Παρατηρώντας το παράδειγμα, καταλαβαίνουμε πως υπάρχουν πολλοί δελεασμοί για να επιλέξει ο καπετάνιος να πάει σε αντίθετη κατεύθυνση από της πατρίδας του για να πάρει το χρυσό, όμως παρόλο που υπάρχουν νησιά που δίνουν ανταμοιβές, υπάρχουν ωστόσο και πολλοί κίνδυνοι, με τους οποίους μπορεί ο πράκτορας να χάσει δύο πόντους και να τελειώσει το παιχνίδι. Άρα, προτιμότερο είναι να μην επιλεχτούν οι αντίθετοι προορισμοί από την πατρίδα, παρόλο που υπάρχουν ανταμοιβές εκεί, και να

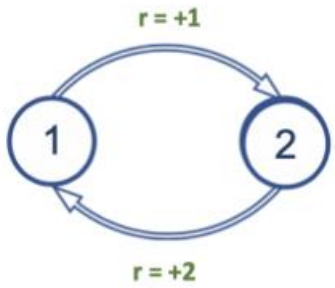
κατευθυνθεί ο πράκτορας στη κατεύθυνση της πατρίδας του που τελικά αυτό μόνο προσδίδει τη μακροπρόθεσμη ανταμοιβή.[40]

4.5 Value iteration

Ο αλγόριθμος επανάληψης τιμών είναι απλός. Συνδυάζει δύο φάσεις της επανάληψης της πολιτικής σε μια ενιαία λειτουργία ενημέρωσης. Ωστόσο, η συνάρτηση επανάληψης τιμής εκτελείται σε όλες τις πιθανές ενέργειες ταυτόχρονα για να βρει τη μέγιστη τιμή ενέργειας.[45] Μια πολιτική $\pi(\alpha|s)$ επιτυγχάνει τη βέλτιστη τιμή από την κατάσταση s, $V_\pi(s) = v*(S)$, αν και μόνο αν για κάθε κατάσταση s' είναι προσβάσιμο από το s και έχει επιτύχει η βέλτιστη τιμή από τη κατάσταση s', , $V_\pi(s') = v*(S')$. Από αυτόν τον ορισμό, μπορούμε να δούμε ότι αυτή η τεχνική χρησιμοποιεί αναδρομή για κάθε κατάσταση προσβάσιμη από το s. Λέει ότι αν γνωρίζουμε τη λύση στα υποπροβλήματα v∗(s′), όπου s′ είναι όλες οι καταστάσεις προσβάσιμες από το s, τότε η λύση v∗(s) μπορεί να βρεθεί με μια απλή ματιά σε μία κατάσταση χρησιμοποιώντας:

$$V_*(s) = \max_{a\varepsilon A} R_s^a + \gamma \sum_{s'\varepsilon S} P_{ss'}^a v*(s')$$

Για παράδειγμα, έστω ότι υπάρχει ένα απλό Περιβάλλον με δύο καταστάσεις, την κατάσταση 1 και την κατάσταση 2, που παρουσιάζει το διάγραμμα μετάβασης καταστάσεων του παρακάτω περιβάλλοντος:



Εικόνα 41: Παράδειγμα καταστάσεων value iteration

Έχουμε μόνο δύο πιθανές μεταβάσεις: από την κατάσταση 1 μπορούμε να κάνουμε μόνο μια ενέργεια που μας οδηγεί στην κατάσταση 2 με Ανταμοιβή +1 και από την κατάσταση 2 μπορούμε να κάνουμε μόνο μια ενέργεια που μας επιστρέφει στην κατάσταση 1 με Ανταμοιβή +2. Έτσι, η ζωή του Πράκτορά μας κινείται σε μια άπειρη ακολουθία καταστάσεων λόγω του άπειρου βρόχου μεταξύ των δύο καταστάσεων. Έστω ότι έχουμε συντελεστή έκπτωσης γ<1, ας πούμε 0,9, και ότι η βέλτιστη τιμή της κατάστασης είναι ίση με αυτή της ενέργειας που μας δίνει τη μέγιστη δυνατή αναμενόμενη άμεση ανταμοιβή, συν την έκπτωση μακροπρόθεσμη ανταμοιβή για την επόμενη κατάσταση:

$$V_*(s) = \max_{a\varepsilon A} \gamma \sum_{s'} P_{ss'}^a \ (r(s,a) + \gamma V*(s'))$$

Στο παράδειγμά μας, καθώς υπάρχει μόνο μία ενέργεια διαθέσιμη σε κάθε κατάσταση, ο Πράκτορας μας δεν έχει άλλη επιλογή και επομένως μπορούμε να απλοποιήσουμε τον προηγούμενο τύπο ως:

$$V_*(s) = (r(s,a) + \gamma V*(s')$$

Για παράδειγμα, αν ξεκινήσουμε από την κατάσταση 1, η ακολουθία των καταστάσεων θα είναι [1,2,1,2,1,2, …], και αφού κάθε μετάβαση από την κατάσταση 1 στην κατάσταση 2 μας δίνει μια ανταμοιβή +1 και κάθε μετάβαση πίσω μας δίνει μια Ανταμοιβή +2, η ακολουθία των Ανταμοιβών θα είναι [+1,+2,+1,+2,+1,+2, …]. Επομένως, ο προηγούμενος τύπος για την κατάσταση 1 γίνεται:

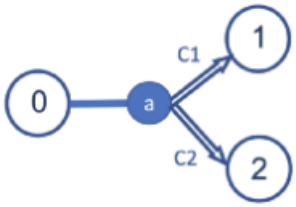
$$v(1) = 1 + \gamma\left(2 + \gamma\big(1 + \gamma(2 + \cdots)\big)\right) = \sum_{i=0}^{\infty} 1\,\gamma^{2i} + 2\gamma^{1+2i}$$

Το προηγούμενο παράδειγμα μπορεί να χρησιμοποιηθεί για να ληφθεί η ουσία μιας πιο γενικής διαδικασίας που ονομάζεται αλγόριθμος Επανάληψης Τιμής (VI). Αυτό μας επιτρέπει να υπολογίσουμε αριθμητικά τις τιμές των καταστάσεων των διαδικασιών απόφασης Markov, με γνωστές πιθανότητες μετάβασης και ανταμοιβές. Η ιδέα πίσω από τον αλγόριθμο επανάληψης τιμής είναι να συγχωνευτεί ένα περικομμένο βήμα αξιολόγησης πολιτικής και μια βελτίωση πολιτικής στον ίδιο αλγόριθμο. Βασικά, ο αλγόριθμος Value Iteration υπολογίζει τη συνάρτηση βέλτιστης τιμής κατάστασης βελτιώνοντας επαναληπτικά την εκτίμηση των V(s). Ο αλγόριθμος αρχικοποιεί τα V(s) σε αυθαίρετες τυχαίες τιμές. Ενημερώνει επανειλημμένα τις τιμές Q(s, a) και V(s) μέχρι να συγκλίνουν. Η επανάληψη τιμής είναι εγγυημένη ότι συγκλίνει στις βέλτιστες τιμές.

Στην πράξη, αυτή η μέθοδος Επανάληψης Τιμής έχει αρκετούς περιορισμούς. Πρώτα απ 'όλα, ο χώρος κατάστασης θα πρέπει να είναι διακριτός και αρκετά μικρός ώστε να εκτελεί πολλαπλές επαναλήψεις σε όλες τις καταστάσεις. Αυτό δεν είναι ένα ζήτημα για το Περιβάλλον μας στην Παγωμένη Λίμνη, αλλά σε ένα γενικό πρόβλημα Εκμάθησης Ενίσχυσης, αυτό δεν ισχύει. Θα αντιμετωπίσουμε αυτό το ζήτημα σε επόμενες αναρτήσεις αυτής της σειράς. Ένα άλλο ουσιαστικό πρακτικό πρόβλημα προκύπτει από το γεγονός ότι για την ενημέρωση της εξίσωσης Bellman, ο αλγόριθμος απαιτεί τη γνώση της πιθανότητας των μεταβάσεων και της Ανταμοιβής για κάθε μετάβαση του Περιβάλλοντος. Όμως, αυτά λύνονται με την εμπειρία του πράκτορα.

Η εκτίμηση των ανταμοιβών και η εκτίμηση των μεταβάσεων είναι το πιο εύκολο μέρος. Οι ανταμοιβές μπορούν να χρησιμοποιηθούν ως έχουν, όμως πρέπει να θυμόμαστε τι ανταμοιβή πήραμε κατά τη μετάβαση από το s στο s χρησιμοποιώντας την ενέργεια α. Επίσης, η εκτίμηση των μεταβάσεων υπολογίζεται με τη χρήση μετρητών για κάθε εμπειρία του πράκτορα.

Για παράδειγμα, μπορούμε να δημιουργήσουμε έναν απλό πίνακα που διατηρεί τους μετρητές των έμπειρων μεταβάσεων. Το κλειδί του πίνακα μπορεί να είναι μια σύνθετη "κατάσταση" + "ενέργεια", (s, a) και οι τιμές κάθε καταχώρισης είναι οι πληροφορίες σχετικά με τις καταστάσεις προορισμού, s' και πλήθος φορών που έχουμε δει το καθένα κατάσταση στόχος, γ. Έστω ότι κατά τη διάρκεια της εμπειρίας του Agent, σε μια δεδομένη κατάσταση s0 έχει εκτελέσει μια ενέργεια πολλές φορές και καταλήγει c1 φορές στην κατάσταση s1 και c2 φορές στην κατάσταση s2. Πόσες φορές έχουμε αλλάξει σε καθεμία από αυτές τις καταστάσεις αποθηκεύεται στον πίνακα μετάβασης.



Εικόνα 42: Παράδειγμα κατανόησης καταστάσεων value iteration

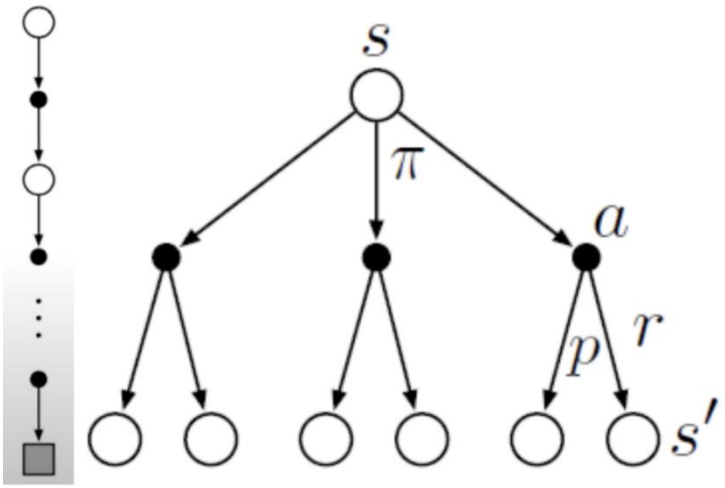
Στη συνέχεια, είναι εύκολο να χρησιμοποιηθεί αυτός ο πίνακας για να υπολογιστούν οι πιθανότητες των μεταβάσεων. Η πιθανότητα ότι η ενέργεια θα οδηγήσει από την κατάσταση 0 στην κατάσταση 1 είναι $\frac{C_1}{C_1+C_2}$ και ότι η ενέργεια θα μας οδηγήσει από την κατάσταση 0 στην κατάσταση 2 $\frac{C_2}{C_1+C_2}$. Για παράδειγμα, έστω ότι από μια κατάσταση 0 εκτελούμε την ενέργεια 1 δέκα φορές, και μετά από 4 φορές θα μας οδηγήσει στην κατάσταση 1 και μετά από 6 φορές θα μας οδηγήσει στην κατάσταση 2. Για το συγκεκριμένο παράδειγμα, η καταχώρηση με το κλειδί (0, 1) στα περιεχόμενα αυτού του πίνακα {1: 4, 2: 6}. Και αυτό αντιπροσωπεύει ότι η πιθανότητα μετάβασης από την κατάσταση 0 στην κατάσταση 1 είναι 4/10, δηλαδή 0,4 και αυτή της κατάστασης 0 στην κατάσταση 2 του 6/10, δηλαδή 0,6. Με αυτές τις πληροφορίες που υπολογίζονται από την εμπειρία του Agent, έχουμε ήδη όλες τις απαραίτητες πληροφορίες για να μπορέσουμε να εφαρμόσουμε τον αλγόριθμο Επανάληψης Τιμής.[45]

4.6 Monte Carlo

Η μέθοδος Monte Carlo περιγράφει τυχαιοποιημένους αλγόριθμους. Οι μέθοδοι Monte Carlo απαιτούν μόνο εμπειρία. Δηλαδή, δειγματίζουν καταστάσεις, ενέργειες και ανταμοιβές, ενώ αλληλοεπιδρούν με το περιβάλλον. Επίσης αποτελεί έναν τρόπο επίλυσης προβλημάτων RL με βάση τη μέση απόδοση δειγμάτων. Μόλις τελειώσει ένα επεισόδιο, αλλάζουν οι εκτιμήσεις αξίας και οι πολιτικές.

Στην ελεύθερη εκμάθηση μοντέλων, δεν έχουμε τη συνάρτηση μετάβασης p(s', r | s, a).Αντίθετα, ενημερώνονται οι καταστάσεις υπολογίζοντας τον μέσο όρο των επιστροφών που βιώνει το μοντέλο σε κάθε κατάσταση, όταν βρίσκεται σε αυτές. Βασικά, μέχρι τώρα γνωρίζαμε όλες τις μεταβάσεις (s → s'), οπότε μπορούσαμε απλώς να ενημερώσουμε μια τιμή κατάστασης υπολογίζοντάς τες. Τώρα, ωστόσο, πρέπει να εξερευνήσουμε πραγματικά, ξεκινώντας από την κατάσταση "s", να δούμε πώς είναι η επόμενη κατάσταση και ενέργεια από την εμπειρία (δηλαδή να δείξουμε μια κατάσταση και μια ενέργεια) και να ενημερώσουμε την τιμή αυτής της κατάστασης "s" υπολογίζοντας τον μέσο όρο των αποτελεσμάτων καθώς εξερευνώ.[47]

$$\bigcup_{k+1}(S) = E_\pi[R_{t+1} + \gamma v_k(S_{t+1})|S_t = s] = \sum_a \pi(a|s) \sum_{s',r} \rho(s', r|s, a)[r + \gamma v_k(s')]$$



Εικόνα 43: Επιστροφές Δειγματοληψίας στα αριστερά και το διάγραμμα στα δεξιά

Στις επιστροφές δειγματοληψίας, ενημερώνουμε την τιμή της κατάστασης "S" με βάση δείγματα επεισοδίων που περνούν από την κατάσταση (αριστερή εικόνα). Συγκριτικά, στο εφεδρικό διάγραμμα πριν κοιτάξαμε 1 βήμα μπροστά σε όλες τις επόμενες καταστάσεις S' και το χρησιμοποιήσαμε για να ενημερώσουμε την κατάσταση S. Εδώ θα δούμε πώς μπορούμε να μάθουμε τη συνάρτηση τιμής κατάστασης για μια πολιτική. Διαφορετικό από τις προηγούμενες μεθόδους RL που βασίζονται σε μοντέλα, είναι ότι εδώ αλλάζουμε τον τρόπο αξιολόγησης της πολιτικής και εκτίμησης των τιμών της κατάστασης δράσης. Επίσης, ακόμη και η τη συνάρτηση τιμής κατάστασης για μια πολιτική υπολογίζεται με διαφορετικό τρόπο από τις προηγούμενες μεθόδους RL που βασίζονται σε μοντέλα, επειδή στο Monte Carlo αλλάζει ο τρόπος αξιολόγησης της πολιτικής και εκτίμησης των τιμών της κατάστασης δράσης. Γενικά όπως είδαμε και σε προηγούμενες μεθόδους, γνωρίζουμε πως η τιμή μιας κατάστασης είναι η αναμενόμενη απόδοση που ξεκινά από αυτήν την κατάσταση και στη συνέχεια ακολουθεί μια συγκεκριμένη πολιτική. Ο μέσος όρος των επιστροφών που βλέπουμε μετά την επίσκεψη σε μια πολιτεία θα ήταν μια εύκολη προσέγγιση για την αξιολόγηση με βάση την εμπειρία. Ο μέσος όρος θα πρέπει να πλησιάζει την προβλεπόμενη τιμή καθώς αλληλοεπιδρούμε περισσότερο με το περιβάλλον και παρατηρούμε περισσότερες αποδόσεις. Όλες οι τεχνικές Monte Carlo λειτουργούν με την ίδια αρχή.

Ας υποθέσουμε ότι θέλουμε να υπολογίσουμε την τιμή μιας κατάστασης $V\pi(s)$. Κάθε φορά που επισκεπτόμαστε την κατάσταση "s" σε ένα επεισόδιο ονομάζεται "επίσκεψη στο s". Σαφώς, μπορεί να επισκεφτούμε το «s» πολλές φορές σε ένα μόνο επεισόδιο. Στο MC πρώτης επίσκεψης, υπολογίζουμε την τιμή της κατάστασης "s" υπολογίζοντας τον μέσο όρο των αποδόσεων που παρατηρούμε μετά την πρώτη μας επίσκεψη στο "s". Επίσης, υπολογίζουμε την τιμή "s", υπολογίζοντας τον μέσο όρο των αποδόσεων μετά από όλες τις επισκέψεις στο "s".

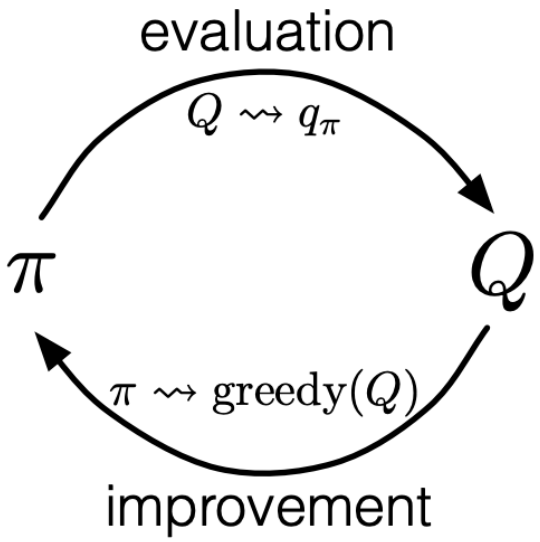
Συνοψίζοντας, αν υπάρξει ένα μοντέλο περιβάλλοντος είναι αρκετά εύκολο να προσδιοριστεί η πολιτική από τις τιμές κατάστασης, όπου εδώ να σημειωθεί πως πάντα πρέπει να κοιτάμε ένα βήμα μπροστά για να δούμε ποια κατάσταση δίνει τον καλύτερο συνδυασμό ανταμοιβής και επόμενης κατάστασης. Όμως αν δεν έχουμε ένα μοντέλο περιβάλλοντος, οι τιμές μιας κατάστασης δεν αρκούν. Σε αυτήν την περίπτωση, είναι χρήσιμο να εκτιμηθούν οι τιμές ενεργειών (οι τιμές των διαφορετικών ενεργειών σε μια κατάσταση) αντί των τιμών κατάστασης. Έτσι, ένας κύριος στόχος των μεθόδων MC είναι να εκτιμήσουν τις βέλτιστες τιμές δράσης $q*$. Για να υπολογιστεί το $q*$, πρώτα πρέπει να γίνει η αξιολόγηση πολιτικής για τις τιμές της δράσης (action values). Αυτό σημαίνει πως θα υπολογιστεί το $q\pi(s, a)$, η αναμενόμενη απόδοση, όταν ξεκινάει το μοντέλο στην κατάσταση "s" και στη συνέχεια θα ακολουθεί μια πολιτική π. Αυτό είναι παρόμοιο με αυτό για το οποίο προαναφέρθηκε με τις τιμές κατάστασης ($V\pi$), με τη διαφορά πως τώρα μιλάμε για επίσκεψη κατάστασης-δράσης και όχι απλώς σε μια κατάσταση.

Μια κατάσταση μπορεί να έχει πολλές ενέργειες. Έτσι, όταν βρίσκεται το μοντέλο σε μια κατάσταση, υπάρχουν πολλές ενέργειες που μπορεί να εκτελέσει. Όμως, στη συγκεκριμένη περίπτωση πρέπει να ληφθεί υπόψιν μας, ότι αναφερόμαστε σε κατάσταση-δράση, εννοώντας πως μιλάμε πάντα για τη λήψη αυτής της συγκεκριμένης δράσης σε αυτήν τη συγκεκριμένη κατάσταση.

Όπως προαναφέρθηκε, στο Monte Carlo, αλλιώς υπάρχει άλλος τρόπους υπολογισμού σε κάθε επίσκεψη και άλλος για την πρώτη επίσκεψη. Στη πρώτη επίσκεψη, πρέπει να υπολογιστεί ο μέσος όρος των αποδόσεων μετά την πρώτη φορά που έγινε αυτή η ενέργεια σε αυτή τη κατάσταση ενώ σε κάθε επίσκεψη πρέπει να υπολογιστεί η αξία της κατάστασης-δράσης, υπολογίζοντας τον μέσο όρο των αποδόσεων που ακολούθησαν τις επισκέψεις σε αυτό.

Το μόνο πρόβλημα είναι πως μπορεί κάποια κατάσταση-φράση να μην έχει γίνει ποτέ. Για παράδειγμα, εάν έχουμε μια ντετερμινιστική πολιτική, θα λάβουμε μόνο μία ενέργεια ανά κράτος (αυτή που ευνοεί η πολιτική). Και ως εκ τούτου, θα παρατηρούμε επιστροφές μόνο για μία ενέργεια. Αυτό είναι το γενικό πρόβλημα της διατήρησης της εξερεύνησης — για να λειτουργήσει η αξιολόγηση της πολιτικής πρέπει να βεβαιωθούμε ότι επισκεπτόμαστε όλες τις δράσεις σε κάθε κράτος. Ένας τρόπος για να λυθεί αυτό το πρόβλημα είναι, να καθοριστεί σε κάθε επεισόδιο κάποια κατάσταση- δράση και ότι κάθε κατάσταση- δράση θα έχει πιθανότητα μεγαλύτερη από μηδέν να επιλεχτεί ως έναρξη. Αυτή η διαδικασία ονομάστηκε ως υπόθεση εξερεύνησης. Αλλά αυτή η υπόθεση δεν λειτουργεί πάντα. Για παράδειγμα, εάν μαθαίνουμε απευθείας από την αλληλεπίδραση με το περιβάλλον, οι συνθήκες εκκίνησης δεν είναι πολύ χρήσιμες. Ένα συνηθισμένος τρόπος είναι να εξετάζονται μόνο οι στοχαστικές πολιτικές όπου η πιθανότητα κάθε δράσης σε κάθε κατάσταση δεν είναι μηδέν.

Η εκτίμηση της μεθόδου του Monte Carlo μπορεί να χρησιμοποιηθεί για έλεγχο, δηλαδή να προσεγγίσει τις βέλτιστες πολιτικές. Η ιδέα είναι να χρησιμοποιηθεί η γενικευμένη επανάληψη πολιτικής (GPI), στην οποία διατηρούμε μια προσέγγιση μιας συνάρτησης πολιτικής και μιας τιμής. Η συνάρτηση τιμής αλλάζει συνεχώς για να είναι καλύτερη προσέγγιση της πολιτικής και η πολιτική βελτιώνεται συνεχώς.



Εικόνα 44: GPI

Όπως προαναφέρθηκε η αξιολόγηση της πολιτικής αξιολογείται με τη κατάσταση-δράση και όχι με τις καταστάσεις. Ενεργώντας σε κάθε κατάσταση, επιτυγχάνεται η συνιστώσα της βελτίωσης της πολιτικής. Δηλαδή, η πολιτική επιλέγει την δράση με την υψηλότερη τιμή δράσης για οποιαδήποτε συνάρτηση τιμής ενέργειας "q" και για οποιαδήποτε κατάσταση "s":

$$\pi(s) = arg\,max_a q(s,a)$$

Για να βεβαιωθούμε ότι όλες οι ενέργειες επιλέγονται απεριόριστα συχνά, πρέπει να τις επιλέγουμε συνεχώς. Υπάρχουν 2 προσεγγίσεις για να διασφαλιστεί αυτό — μέθοδοι εντός πολιτικής και μέθοδοι εκτός πολιτικής. Η μια προσέγγιση είναι οι μέθοδοι επί της πολιτικής (On-policy methods), όπου γίνεται προσπάθεια αξιολόγησης ή βελτίωση της πολιτικής. Η άλλη προσέγγιση είναι οι μέθοδοι εκτός πολιτικής (Off-policy methods), όπου υπάρχουν δύο πολιτικές και πρέπει να αξιολογηθεί ή να βελτιωθεί μία από αυτές και να χρησιμοποιηθεί η άλλη για οδηγίες.

Για να γίνει πιο ξεκάθαρο όσον αφορά την πρόβλεψη εκτός πολιτικής (Off-policy), έστω ότι υπάρχουν επεισόδια που δημιουργούνται από διαφορετική πολιτική, δηλαδή μέθοδο εκτός πολιτικής. Ο τρόπος με τον οποίο ακολουθούμε αυτό είναι διατηρώντας μια πολιτική στόχου π, η οποία είναι η πολιτική που θα προσπαθήσει να συμπεριφέρεται βέλτιστα, και θα έχουμε επίσης μια πολιτική συμπεριφοράς b, η οποία είναι η πολιτική εξερεύνησής μας. Η ιδέα είναι ότι θα χρησιμοποιήσουμε επεισόδια που δημιουργούνται από την πολιτική συμπεριφοράς για να εξερευνήσουμε το περιβάλλον και στη συνέχεια να το χρησιμοποιήσουμε για να ενημερώσουμε την πολιτική στόχου π. Το κάνουμε αυτό για να υπολογίσουμε Vπ ή qπ.[47] Για να λειτουργήσει αυτή η μέθοδος, η οποία είναι μια μέθοδος εκτίμησης των αναμενόμενων τιμών για μια κατανομή, δεδομένων δειγμάτων από μια άλλη κατανομή, πρέπει να ευθυγραμμίσουμε τα π και b με τη δειγματοληψία σπουδαιότητας. Βασικά, θα σταθμίσουμε τις αποδόσεις με βάση τη σχετική πιθανότητα οι τροχιές τους να συμβούν κάτω από τα π και b. Αυτό το ονομάζουμε αναλογία σπουδαιότητας-δειγματοληψίας.

A1	0.2
A2	0.6
A3	0.1
A4	0.1

A1	0.7
A2	0.1
A3	0.1
A4	0.1

Εικόνα 45: π (αριστερά) και β (δεξιά)

Έστω ότι υπάρχουν τα 2 παραπάνω διανύσματα της φωτογραφίας. Βλέποντας την πολιτική στόχου π, βλέπουμε ότι έχουμε 20% πιθανότητα να προβούμε σε ενέργεια A1, όπου η πολιτική συμπεριφοράς b έχει 70% πιθανότητα να λάβει την A1. Αυτού του είδους οι διαφορετικές πιθανότητες για ενέργειες είναι ο λόγος που πρέπει να σταθμίσουμε τις ενέργειες διαφορετικά στο π (αριστερά) και στο b (δεξιά). Η δειγματοληψία για την ευθυγράμμιση των πιθανοτήτων των δύο πολιτικών γίνεται:

$$p_{t:}T-1=\frac{\Pi_{k=t}^{T-1}\pi(A_k|s_k)p(S_{k+1}|S_k,A_k)}{\Pi_{k=t}^{T-1}b(A_k|s_k)p(S_{k+1}|S_k,A_k)}=\prod_{k=t}^{T-1}\frac{\pi(A_k|s_k)}{b(A_k|s_k)}$$

Εξετάζοντας τον τύπο για την αναλογία σπουδαιότητας-δειγματοληψίας, , είναι το προϊόν της λήψης μιας συγκεκριμένης ενέργειας δεδομένης μιας συγκεκριμένης κατάστασης από την πολιτική στόχου σε σύγκριση με την πολιτική συμπεριφοράς. Και πάλι, το κάνουμε αυτό για να μπορούμε να κάνουμε μάθηση εκτός πολιτικής. Ειδικότερα, η πολιτική συμπεριφοράς βιώνει το περιβάλλον και λαμβάνει μια επιστροφή Gt = 10. Εάν η συμπεριφορά που πήρε το b σε μια συγκεκριμένη κατάσταση είναι λιγότερο

πιθανό να συμβεί από ό,τι είναι στο π, ας πούμε 3, θα ζυγίζαμε την απόδοση κατά 3 και θα έχουμε την πραγματική απόδοση όταν ενημερώνουμε το π να είναι 30:

$$\pi(a|s) > b(a|s) \rightarrow p > 1 \text{ π.χ. } 3 \rightarrow Gt = 10 \cdot 3 = 30$$

Όμως, εάν η συμπεριφορά είναι πολύ πιο πιθανό να συμβεί στο b παρά στο π, δεν θα σταθμίσουμε την απόδοση τόσο πολύ κατά την ενημέρωση του π:

$$\pi(a|s) > b(a|s) \rightarrow p > 1 \text{ π.χ. } 0.25 \rightarrow Gt = 10 \cdot 0.25 = 2.5$$

4.7 Temporal Difference learning

Μεταξύ των μεθόδων χωρίς μοντέλα της RL είναι η μάθηση με χρονική διαφορά (temporal difference TD learning). Η εκμάθηση TD είναι μια τεχνική χωρίς επίβλεψη για την πρόβλεψη της αναμενόμενης τιμής μιας μεταβλητής σε μια ακολουθία καταστάσεων. Το TD χρησιμοποιεί ένα μαθηματικό τέχνασμα για να αντικαταστήσει τη σύνθετη λογική για το μέλλον με μια απλή διαδικασία μάθησης που μπορεί να παράγει τα ίδια αποτελέσματα. Αντί να υπολογίσει τη συνολική μελλοντική ανταμοιβή, η TD προσπαθεί να προβλέψει τον συνδυασμό της άμεσης ανταμοιβής και της δικής της πρόβλεψης ανταμοιβής την επόμενη χρονική στιγμή.

$$R_t = r_{t+1} + r_{t+2} + \gamma^2 r_{r+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{r+k+1}$$

Όπου η ανταμοιβή τη στιγμή t είναι ο συνδυασμός ανταμοιβών με έκπτωση στο μέλλον. Υπονοεί ότι οι μελλοντικές ανταμοιβές αποτιμώνται λιγότερο. Το Σφάλμα TD είναι η διαφορά μεταξύ της τελικής σωστής ανταμοιβής και της τρέχουσας πρόβλεψής μας.

Επιπλέον, αυτή η μέθοδος χρησιμοποιεί συγκεκριμένους παραμέτρους. Ειδικότερα, μια παράμετρος αποτελεί το Alpha (α), όπου αντιπροσωπεύει το ρυθμό μεταβολής και δείχνει πόσο πρέπει να προσαρμόσουμε τις εκτιμήσεις μας με βάση το σφάλμα. Ο ρυθμός εκμάθησης είναι μεταξύ 0 και 1 και μάλιστα ένας μεγάλος ρυθμός μάθησης προσαρμόζεται γρήγορα και μπορεί να οδηγήσει σε κυμαινόμενα αποτελέσματα εκπαίδευσης — όχι σε σύγκλιση. Αντίθετα, ένας μικρός ρυθμός μάθησης προσαρμόζεται αργά, κάτι που θα χρειαστεί περισσότερο χρόνο για να συγκλίνει. Επιπλέον, το Gamma (γ) αποτελεί άλλη μια παράμετρο της μεθόδου, όπου αντιπροσωπεύει το ποσοστό έκπτωσης (discount rate) και δείχνει την εκτίμηση των μελλοντικών ανταμοιβών. Τέλος, μια ακόμη παράμετρος αποτελεί το e-greedy όπου αντικατοπτρίζει την εξερεύνηση ενάντια της εκμετάλλευσης. Δηλαδή, εξερευνά νέες επιλογές με πιθανότητα e και το e συνεπάγεται με περισσότερη εξερεύνηση κατά την προπόνηση. [49]

Τέλος, τέσσερις από τους πιο χρησιμοποιούμενους αλγόριθμους της TD αποτελούν οι SARSA, Q-learning (QL), η Expected Sarsa learning και ο Double Q-Learning, οι οποίοι θα αναλυθούν στα παρακάτω κεφάλαια.

4.7.1 Q-learning & Sarsa & Expected Sarsa

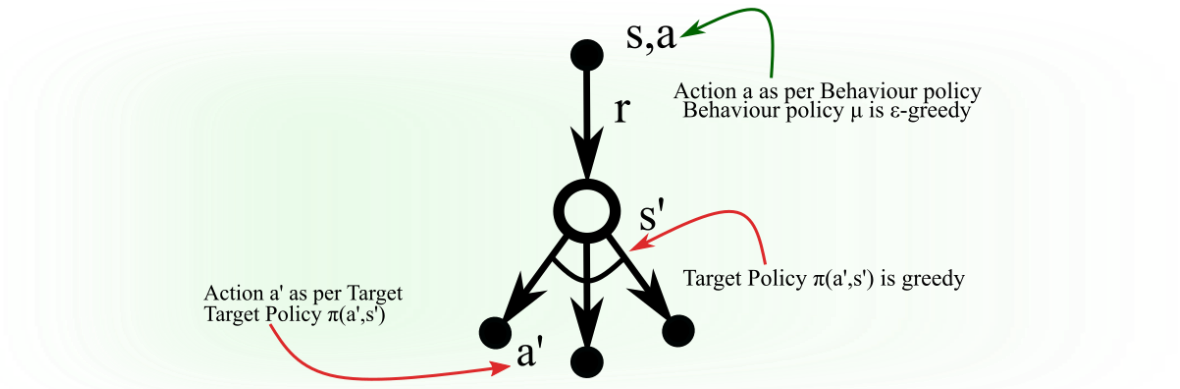
Το QL και το SARSA αποθηκεύουν τις ανταμοιβές της τρέχουσας κατάστασης και την αντίστοιχη ενέργεια για μελλοντική ενημέρωση. Μια κατάσταση συνήθως αντιπροσωπεύεται από τις συντεταγμένες των συστατικών της. Αν το περιβάλλον είναι συνεχές, θα υπάρξει άπειρος αριθμός καταστάσεων. Για να αντιμετωπιστεί αυτό το πρόβλημα, πρέπει να γίνει διακριτοποίηση τμημάτων ης κατάστασης. Για παράδειγμα, έστω ότι υπάρχει ένας συνεχής χώρος, ο οποίος είναι χωρισμένος σε πλέγματα και έστω ότι υπάρχει μια ξεχωριστή συνάρτηση για τη διακριτοποίηση τους, τέσσερις

μεταβλητές, οι οποίες είναι χωρισμένες σε δέκα κουτιά το καθένα. Άρα, το βάρος του πίνακα είναι 10^4. Όσο περισσότερα κουτιά υπάρχουν, τόσο περισσότερες λεπτομέρειες μπορεί να μάθει το μοντέλο, όμως αυτό σημαίνει ότι θα υπάρξει περισσότερος χρόνος για εκμάθηση και περισσότερος χώρος στη μνήμη. Αντίθετα, εάν ο αριθμός των κουτιών είναι πολύ μικρός, το μοντέλο θα έχει χαμηλή απόδοση.

Καθώς χρειάζεται να απομνημονεύσει τον χώρο κατάστασης, το QL και το SARSA θα λειτουργήσουν καλύτερα για παιχνίδια περιορισμένων και περιορισμένων ενεργειών (όπως το CartPole που κινείται αριστερά και δεξιά) και όχι καλά σε πιο σύνθετα (όπως το σκάκι με πολλές πιθανές κινήσεις). Για να αποφύγουμε την αποθήκευση όλου του χώρου κατάστασης για παιχνίδια με μεγαλύτερο χώρο κατάστασης, μπορούμε να χρησιμοποιήσουμε ένα deep q δίκτυο. Αυτή η προσέγγιση συνδυάζει την ενισχυτική μάθηση με τα νευρωνικά δίκτυα. Όπως προαναφέρθηκε, η πολιτική είναι μια στρατηγική που χρησιμοποιεί ένας πράκτορας για να επιδιώξει έναν στόχο. Το Greedy (η επιλογή της καλύτερης αξίας) είναι πολιτική. Ο αλγόριθμος greedy αποτελεί μια πολιτική που ζητά την ευκαιρία να εξερευνήσει και να ακολουθήσει τη βέλτιστη διαδρομή με (1-e) πιθανότητα. Τέλος, αυτός ο αλγόριθμος εφαρμόζεται για να εξισορροπήσει την εξερεύνηση και την εξερεύνηση της ενισχυτικής μάθησης.[49][48]

Το QL είναι ο χώρος αποθήκευσης για τη διατήρηση όλων των ανταμοιβών σε διακριτό χώρο και δράση, το s αντιπροσωπεύει την κατάσταση και το a αντιπροσωπεύει την δράση. Στο Q-Learning, ο πράκτορας μαθαίνει τη βέλτιστη πολιτική με τη βοήθεια μιας άπληστης πολιτικής και συμπεριφέρεται χρησιμοποιώντας πολιτικές άλλων πρακτόρων. Το Q-learning αποτελεί μια εκτός πολιτικής (off-policy) ενισχυτικής μάθησης επειδή η ενημερωμένη πολιτική είναι διαφορετική από την πολιτική συμπεριφοράς, επομένως το Q-Learning είναι εκτός πολιτικής. [49] Στην εκμάθηση εκτός πολιτικής, αξιολογούμε την πολιτική στόχου (π) ενώ ακολουθούμε μια άλλη πολιτική που ονομάζεται πολιτική συμπεριφοράς (μ).[52]Με άλλα λόγια, εκτιμά την ανταμοιβή για μελλοντικές ενέργειες και προσθέτει μια αξία στο νέο κράτος χωρίς να ακολουθεί ουσιαστικά καμία άπληστη πολιτική. Στην Q-learning, η πολιτική στόχου είναι μια άπληστη πολιτική και η πολιτική συμπεριφοράς είναι η πολιτική ε-άπληστη, το οποίο διασφαλίζει την εξερεύνηση.

$$Q(S_t, A_t) = Q(S_t, A_t) + a[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \text{ [49]}$$

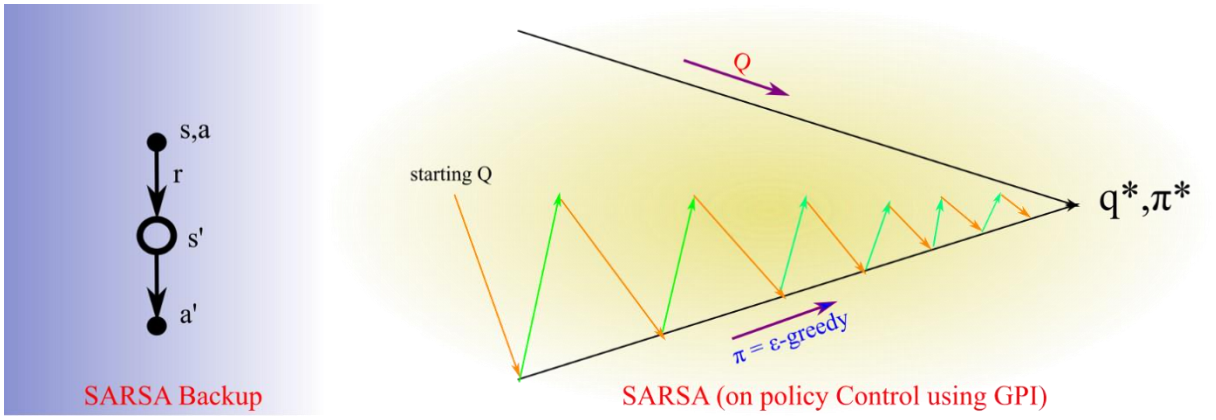


Εικόνα 46: QL- πολιτική

Όσον αφορά τη SARSA (State–action–reward–state–action) αποτελεί έναν αλγόριθμο εκμάθησης ενίσχυσης πολιτικής (on-policy) που εκτιμά την αξία της πολιτικής που ακολουθείται. Σε αυτόν τον

αλγόριθμο, ο πράκτορας κατανοεί τη βέλτιστη πολιτική και χρησιμοποιεί την ίδια για να ενεργήσει. Η πολιτική που χρησιμοποιείται για την ενημέρωση και η πολιτική που χρησιμοποιείται για την ενεργοποίηση είναι η ίδια, σε αντίθεση με το Q-learning.[48] Το SARSA χρησιμοποιεί τη συνάρτηση δράσης-τιμής Q και ακολουθεί την πολιτική π.. Το GPI (Επανάληψη γενικευμένης πολιτικής) χρησιμοποιείται για την ανάληψη δράσης με βάση την πολιτική π (ε-άπληστο για τη διασφάλιση της εξερεύνησης καθώς και για τη βελτίωση της πολιτικής).[52]

$$Q(S_t, A_t) = Q(S_t, A_t) + a[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)][48]$$



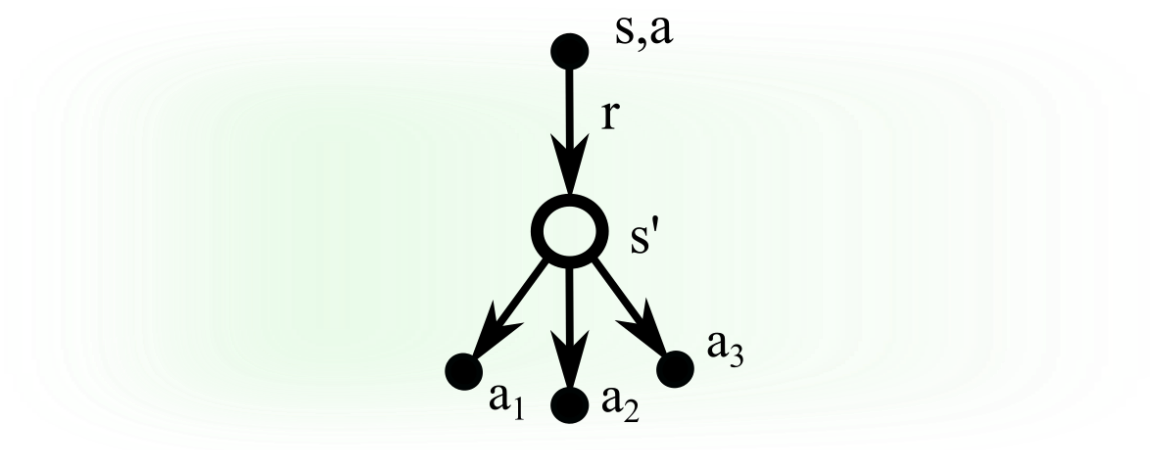
Εικόνα 47: Sarsa - πολιτική

Επιπροσθέτως, υπάρχουν κιάλες διαφορές ανάμεσα σε αυτούς τους δύο αλγορίθμους. Ειδικότερα, στο QL, όταν μεταβιβάζεται η ανταμοιβή από την επόμενη κατάσταση στην τρέχουσα κατάσταση, παίρνει τη μέγιστη δυνατή ανταμοιβή της νέας κατάστασης και αγνοεί όποια πολιτική κι αν χρησιμοποιούμε. Αντίθετα, στο SARSA, εξακολουθούμε να ακολουθούμε την πολιτική (e-greedy), υπολογίζοντας την επόμενη κατάσταση και μεταφέρουμε την ανταμοιβή που αντιστοιχεί στο προηγούμενο βήμα. Συνοπτικά, το QL εξετάζει την καλύτερη δυνατή περίπτωση εάν φτάσετε στην επόμενη κατάσταση, ενώ η SARSA εξετάζει την ανταμοιβή εάν ακολουθήσουμε την τρέχουσα πολιτική στην επόμενη κατάσταση. Ως εκ τούτου, εάν η πολιτική μας είναι άπληστη (greedy), το SARSA και το QL θα είναι το ίδιο αλλά χρησιμοποιούμε το e-greedy εδώ, οπότε υπάρχει μια μικρή διαφορά.

Η Expected SARSA είναι μια εναλλακτική λύση για τη βελτίωση της πολιτικής του πράκτορα. Είναι πολύ παρόμοιο με το SARSA και το Q-Learning αλλά διαφέρει στη συνάρτηση τιμής δράσης που ακολουθεί. Αυτή η μέθοδος μπορεί να χρησιμοποιηθεί είτε ως εντός πολιτικής είτε ως εκτός πολιτικής. Είναι πιο ευέλικτο σε σύγκριση με τους δύο παραπάνω αλγόριθμους.

$$Q(S_t, A_t) = Q(S_t, A_t) + a[R_{t+1} + \gamma \Sigma_a \pi(a|\, S_{t+1}) Q(S_{t+1} A) - Q(S_t, A_t)]$$

Παρατηρώντας τον τύπο, καταλαβαίνουμε πως το Expected SARSA λαμβάνει το σταθμισμένο άθροισμα όλων των πιθανών επόμενων ενεργειών σε σχέση με την πιθανότητα να γίνει αυτή η ενέργεια. Εάν η αναμενόμενη απόδοση είναι άπληστη σε σχέση με την αναμενόμενη απόδοση, τότε αυτή η εξίσωση μετατρέπεται σε Q-Learning. Διαφορετικά, το Expected SARSA είναι εντός πολιτικής και υπολογίζει την αναμενόμενη απόδοση για όλες τις ενέργειες, αντί να επιλέγει τυχαία μια ενέργεια όπως το SARSA.[50]

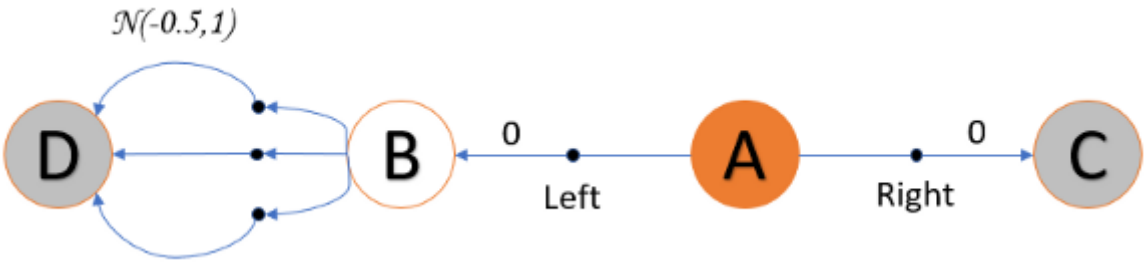


Εικόνα 48: Expected SARSA - πολιτική [52]

4.8 Double Q-Learning

Η Q-learning (Watkins, 1989) θεωρείται μία από τις ανακαλύψεις στον αλγόριθμο μάθησης ενίσχυσης ελέγχου TD. Ωστόσο, ο Hado van Hasselt εξηγεί πώς το Q-Learning έχει πολύ κακή απόδοση σε ορισμένα στοχαστικά περιβάλλοντα. Επισήμανε ότι η κακή απόδοση προκαλείται από μεγάλη υπερεκτίμηση των τιμών δράσης λόγω της χρήσης του Max Q(s',a) στην Q-learning. Για να λύσει αυτό το πρόβλημα πρότεινε τη μέθοδο Double Q-Learning.

Έστω ότι υπάρχει ένα MDP που έχει τέσσερις καταστάσεις από τις οποίες οι δύο είναι τερματικές καταστάσεις. Η κατάσταση A θεωρείται πάντα στην κατάσταση έναρξης και έχει δύο ενέργειες, είτε Δεξιά είτε Αριστερά. Η ενέργεια Right δίνει μηδενική ανταμοιβή και προσγειώνεται στην τερματική κατάσταση C. Η ενέργεια Αριστερά μετακινεί τον πράκτορα στην κατάσταση B με μηδενική ανταμοιβή.



Εικόνα 49: Παράδειγμα κατανόησης Double Q-Learning

Η κατάσταση B έχει έναν αριθμό ενεργειών, οι οποίοι μετακινούν τον πράκτορα στην τερματική κατάσταση D. Ωστόσο η ανταμοιβή R κάθε ενέργειας από το B στο D έχει μια τυχαία τιμή που ακολουθεί μια κανονική κατανομή με μέσο όρο -0,5 και μια διακύμανση 1.0. Η αναμενόμενη τιμή του R είναι γνωστό ότι είναι αρνητική (-0,5). Αυτό σημαίνει ότι σε μεγάλο αριθμό πειραμάτων η μέση τιμή του R είναι μικρότερη από το μηδέν. Με βάση αυτή την υπόθεση, είναι σαφές ότι η μετακίνηση αριστερά από το A είναι πάντα κακή ιδέα. Ωστόσο, επειδή ορισμένες από τις τιμές του R είναι θετικές, το Q-Learning θα εξαπατηθεί για να θεωρήσει ότι η μετακίνηση προς τα αριστερά από το A μεγιστοποιεί την ανταμοιβή. Στην πραγματικότητα αυτή είναι μια κακή απόφαση, γιατί ακόμα κι αν λειτουργήσει για ορισμένα επεισόδια, μακροπρόθεσμα είναι εγγυημένα μια αρνητική ανταμοιβή.

Έστω X1 και X2 δύο τυχαίες μεταβλητές που αντιπροσωπεύουν την ανταμοιβή δύο ενεργειών στην κατάσταση B. Δεδομένου ότι είναι τυχαίες μεταβλητές, θα υπολογίσουμε τις αναμενόμενες τιμές τους E(X1) και E(X2). Ωστόσο, υπάρχει ένα πρόβλημα, δεν γνωρίζουμε τις αναμενόμενες τιμές τους, επομένως αυτό που μπορούμε να κάνουμε είναι να χρησιμοποιήσουμε εκτιμήσεις αυτών των αναμενόμενων τιμών υπολογίζοντας τον αυξητικό μέσο όρο $\boldsymbol{\mu}1$ και $\boldsymbol{\mu}2$. Αυτές οι εκτιμήσεις είναι αμερόληπτες, επειδή όσο αυξάνεται ο αριθμός των δειγμάτων, ο μέσος όρος για ολόκληρο το σύνολο τιμών πλησιάζει περισσότερο στο E(X1) και το E(X2), όπως φαίνεται στην παρακάτω εικόνα.

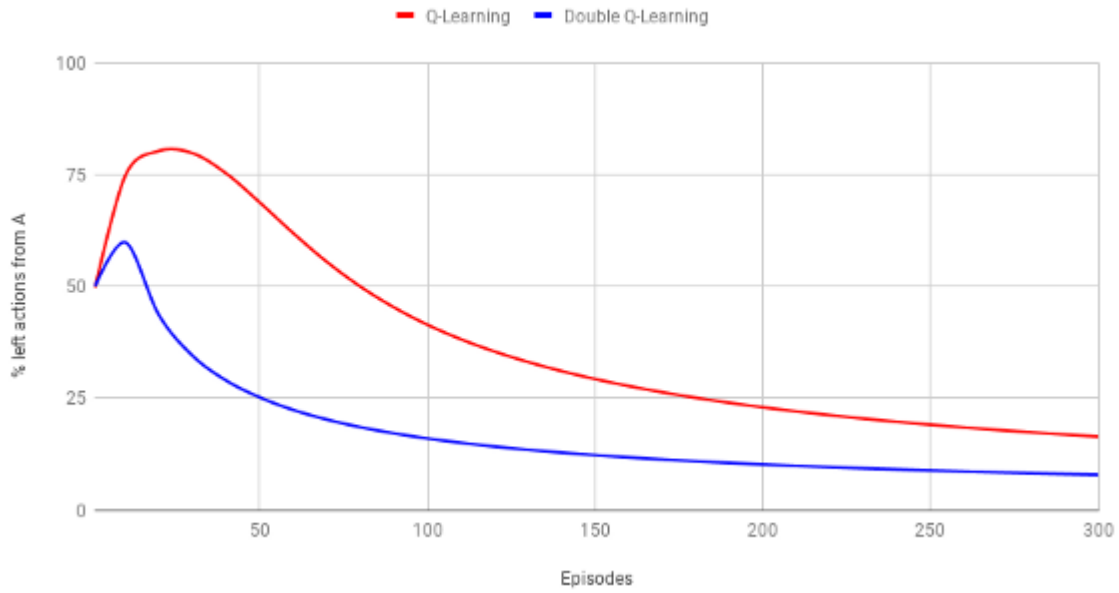
Steps	X1	μ1		X2	μ2	Max(μ)
1	3	3.00		-8	-8.00	3.00
2	-2	0.50		-5	-6.50	0.50
3	4	1.67		-8	-7.00	1.67
4	-1	1.00		-10	-7.75	1.00
5	-5	-0.20		5	-5.20	-0.20
6	2	0.17		-7	-5.50	0.17
7	-6	-0.71		-3	-5.14	-0.71
8	-3	-1.00		3	-4.13	-1.00
9	8	0.00		8	-2.78	0.00
10	-6	-0.60		9	-1.60	-0.60
11	-9	-1.36		2	-1.27	-1.27
12	9	-0.50		-1	-1.25	-0.50
13	1	-0.38		2	-1.00	-0.38
14	-1	-0.43		8	-0.36	-0.36
E(X)	-0.43			-0.36	E(Max(μ))	0.09
Max E(X)	-0.36					

Εικόνα 50: Εκτιμήσεις που αντιπροσωπεύουν το πρόβλημα του Double Q-Learning

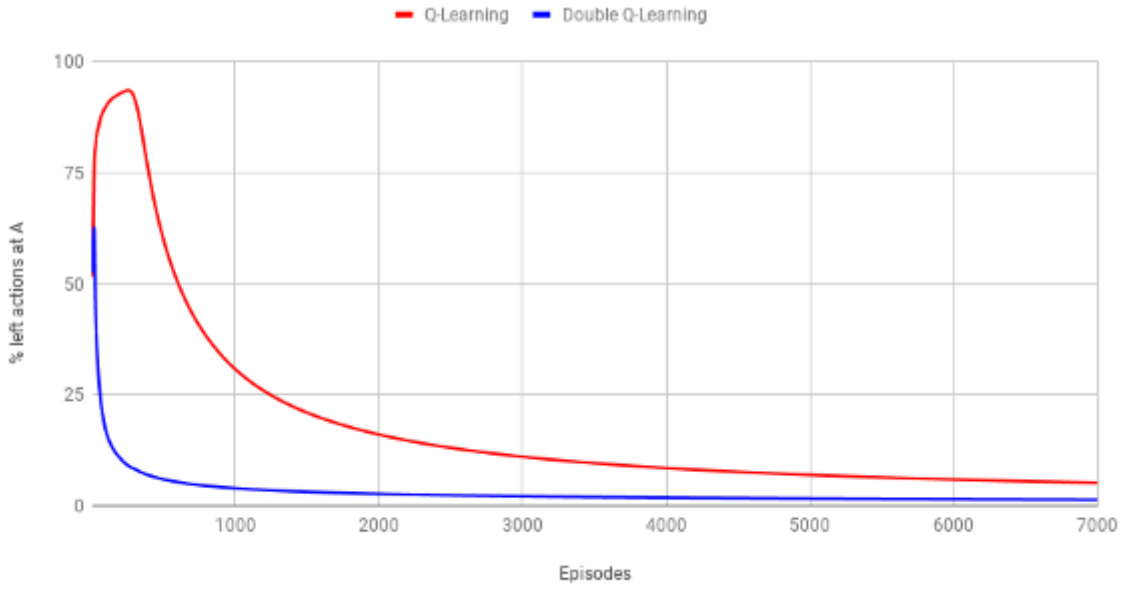
Ωστόσο, το Q-Learning χρησιμοποιεί το Max Q(s’,a), που αντιπροσωπεύεται στον πίνακα από το Max($\boldsymbol{\mu}$). Μπορεί να φανεί ξεκάθαρα από τον πίνακα (βλ. ερυθρά αιμοσφαίρια) ότι το E(Max($\boldsymbol{\mu}$)) είναι διαφορετικό από το Max E(X). Αυτό δείχνει ότι το Max($\boldsymbol{\mu}$) δεν είναι καλός εκτιμητής για το Max E(X). Με άλλα λόγια, κατά την ενημέρωση του Q(s,a) με το Max Q(s',a), το Q(s,a) δεν κινείται προς την αναμενόμενη τιμή των ενεργειών στην κατάσταση B που είναι -0,5.Αυτό το σενάριο δίνει μια απάντηση για το γιατι η Q-Learning υπερεκτιμαται

Η προτεινόμενη λύση είναι η διατήρηση δύο συναρτήσεων Q-value QA και QB, καθεμία από τις οποίες ενημερώνεται από την άλλη για την επόμενη κατάσταση. Η ενημέρωση συνίσταται στην εύρεση της ενέργειας a* που μεγιστοποιεί το QA στην επόμενη κατάσταση (Q(s', a*) = Max Q(s', a)), και στη συνέχεια χρησιμοποείται το a* για να ληγθεί η τιμή των QB(s', α*) προκειμένου να ενημερωθεί το QA(s, a).

Τα παρακάτω γραφήματα δείχνουν μια σύγκριση μεταξύ Double Q-Learning και Q-Learning όταν ο αριθμός των ενεργειών στην κατάσταση B είναι 10 και 100 διαδοχικά. Είναι σαφές ότι το Double Q-Learning συγκλίνει πιο γρήγορα από το Q-learning. Παρατηρήστε ότι όταν ο αριθμός των ενεργειών στο B αυξάνεται, το Q-learning χρειάζεται πολύ περισσότερη εκπαίδευση από το Double Q-Learning.



Εικόνα 51: Performance Double Q-Learning vs Q-Learning (1)



Εικόνα 52: Performance Double Q-Learning vs Q-Learning (2)

Ο Van Hasselt αποδεικνύει στην εργασία του ότι $E(Q2(s', a^*)) \leq \text{Max } Q1(s', a^*)$.Έτσι, σε αρκετό αριθμό πειραμάτων, η αναμενόμενη τιμή του Q2(s', a*) είναι μικρότερη ή ίση με το Max Q1(s', a*), πράγμα που σημαίνει ότι το Q1(s, a) δεν ενημερώνεται με μέγιστη τιμή.

Η παρακάτω εικόνα δείχνει την εξέλιξη της δράσης Q-Values της Αριστεράς στην κατάσταση A καθώς ο αριθμός των επεισοδίων αυξάνεται. Παρατηρήστε ότι στο Q-Learning, το Q(A, Left) είναι θετικό επειδή επηρεάζεται από τις θετικές ανταμοιβές που υπάρχουν στην κατάσταση B. Λόγω αυτής της θετικής τιμής, ο αλγόριθμος ενδιαφέρεται περισσότερο να κάνει την ενέργεια Left ελπίζοντας να μεγιστοποιήσει τις ανταμοιβές. Όπως μπορείτε να δείτε, το ποσοστό της αριστερής δράσης συνεχίζει να αυξάνεται μέχρι το 50ο επεισόδιο. Στο Double Q-Learning Q1(A, Left) και Q2(A, Left) ξεκινούν ελαφρώς αρνητικά. Ως αποτέλεσμα, το ποσοστό της αριστερής δράσης αρχίζει να μειώνεται πολύ νωρίς, εξοικονομώντας έτσι χρόνο προπόνησης.

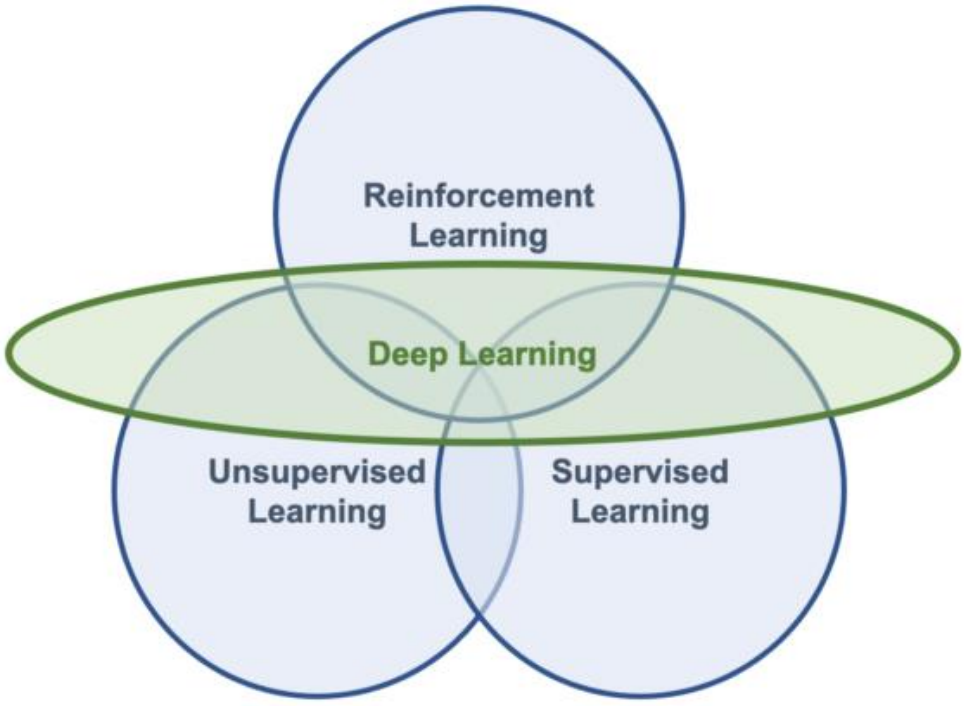
	Q-Learning		Double Q-Learning		
Episodes	% Action left from A	Q(A, Left)	% Action left from A	Q1(A, Left)	Q2(A, Left)
1	50	0	50	0	0
10	75	0.02841949847	60	-0.008121592391	-0.008086578471
20	80	0.01253210403	44	-0.006449018817	-0.006437521554
30	80	0.00500216453	35	-0.005225760654	-0.005192413624
40	75	0.001544030838	29	-0.004329732244	-0.004332900635
50	69	0.0001729521663	25	-0.003734652891	-0.003742598386
60	62	-0.0003455247988	22	-0.003272743797	-0.003284997472
70	56	-0.0005369030957	20	-0.002923117818	-0.002939421609
80	50	-0.0006058136206	19	-0.002638038169	-0.002658186936
90	45	-0.0006310891787	17	-0.002415594122	-0.002433305324
100	41	-0.0006303885304	16	-0.002230909245	-0.002248375734
110	38	-0.000624055014	15	-0.002074685857	-0.002094677605
120	35	-0.0006148598156	14	-0.001936347084	-0.001957402257
130	33	-0.0006037537278	14	-0.001817357216	-0.001839515524
140	31	-0.000590928277	13	-0.001718139458	-0.001736943042
150	29	-0.0005781101555	12	-0.001628659073	-0.00164682604
160	28	-0.0005647542283	12	-0.001550225545	-0.001564606393
170	26	-0.0005531064098	11	-0.0014760005	-0.001493502563
180	25	-0.0005408051867	11	-0.001411351995	-0.001429342003
190	24	-0.0005293939925	11	-0.001354756512	-0.001369770551
200	23	-0.0005182191458	10	-0.001299998376	-0.001316214478
210	22	-0.0005075878904	10	-0.00125025127	-0.001266546273
220	21	-0.0004965378484	10	-0.001205069411	-0.001217966028
230	20	-0.0004865496829	9	-0.001164595728	-0.0011778759
240	20	-0.0004770008354	9	-0.001126632103	-0.001138762389
250	19	-0.0004678410596	9	-0.001089842998	-0.001102551711
260	19	-0.0004587442131	9	-0.001055676975	-0.001067898416
270	18	-0.0004504752567	8	-0.001024680432	-0.00103602158
280	17	-0.0004423669349	8	-0.0009963873672	-0.001005211548
290	17	-0.0004344996438	8	-0.0009689796268	-0.0009770446643
300	16	-0.0004269773881	8	-0.0009434433062	-0.0009499355379

Εικόνα 53: Εξέλιξη δράσης Q-Values και αριθμός επεισοδίων

Κεφάλαιο 5ο: Βαθιά Ενισχυτική Μάθηση

Η μάθηση βαθιάς ενίσχυσης συνδυάζει τεχνητά νευρωνικά δίκτυα με ένα πλαίσιο ενισχυτικής μάθησης που βοηθά τους πράκτορες λογισμικού να μάθουν πώς να επιτύχουν τους στόχους τους. Με άλλα λόγια, συνδυάζει τη βελτιστοποίηση στόχων με την προσέγγιση συναρτήσεων, συνδέοντας καταστάσεις και συμπεριφορές με τις ανταμοιβές που παρέχουν. Ενώ τα νευρωνικά δίκτυα είναι υπεύθυνα για τις πρόσφατες ανακαλύψεις της τεχνητής νοημοσύνης σε προβλήματα όπως η όραση υπολογιστών, η μηχανική μετάφραση και η πρόβλεψη χρονοσειρών, μπορούν να συνδυαστούν με αλγόριθμους ενισχυτικής μάθησης για να δημιουργήσουν κάτι εκπληκτικό όπως παγκόσμιους αθλητές να κερδίζον σε κάποιον αγώνα. [53] Γι' αυτό είναι υπεύθυνο το deep RL, το οποίο είναι μια προσέγγιση της μηχανικής μάθησης και οι αλγόριθμοι του βασίζονται σε τεχνητά νευρωνικά δίκτυα, των οποίων οι αλγοριθμικές δομές επιτρέπουν σε μοντέλα που αποτελούνται από πολλαπλά επίπεδα επεξεργασίας να μαθαίνουν αναπαραστάσεις δεδομένων με διάφορα επίπεδα αφαίρεσης. [54]

Το DL δεν είναι ξεχωριστός κλάδος ML, επομένως δεν είναι διαφορετική εργασία από αυτές που περιγράφονται παραπάνω. Το DL είναι μια συλλογή τεχνικών και μεθόδων για τη χρήση νευρωνικών δικτύων για την επίλυση εργασιών ML, είτε εποπτευόμενης μάθησης, μη εποπτευόμενης μάθησης ή μάθησης ενίσχυσης.



Εικόνα 54: Οπτική σχέση μεταξύ των κατηγοριών Deep Learning και Machine Learning

Το Deep Learning είναι ένα από τα καλύτερα εργαλεία που έχουμε σήμερα για να χειριστούμε μη δομημένα περιβάλλοντα. μπορούν να μάθουν από μεγάλες ποσότητες δεδομένων ή να ανακαλύψουν μοτίβα. Αλλά αυτό δεν είναι λήψη αποφάσεων αλλά είναι πρόβλημα αναγνώρισης. Το Reinforcement Learning παρέχει αυτή τη δυνατότητα.

Η ενισχυτική μάθηση αναφέρεται σε αλγόριθμους προσανατολισμένους στο στόχο, οι οποίοι μαθαίνουν πώς να επιτυγχάνουν έναν σύνθετο στόχο (στόχο) ή πώς να μεγιστοποιούν κατά μήκος μιας συγκεκριμένης διάστασης σε πολλά βήματα. για παράδειγμα, μπορούν να μεγιστοποιήσουν τους πόντους που κερδίζουν σε ένα παιχνίδι σε πολλές κινήσεις. Όμως, οι αλγόριθμοι ενισχυτικής μάθησης που ενσωματώνουν βαθιά νευρωνικά δίκτυα είναι ικανοί να νικήσουν κάποιον αντίπαλο. Αν και αυτό μπορεί να ακούγεται απλό, είναι μια τεράστια βελτίωση σε σχέση με τα προηγούμενα επιτεύγματα της ενισχυτικής μάθησης και η κατάσταση του τομέα προχωρά με ταχείς ρυθμούς. Επίσης, η ενισχυτική μάθηση λύνει το δύσκολο πρόβλημα του συσχετισμού των άμεσων ενεργειών με τα καθυστερημένα αποτελέσματα που παράγουν. Όπως για παράδειγμα, οι άνθρωποι, έτσι και οι αλγόριθμοι ενισχυτικής μάθησης, μερικές φορές καθυστερούν να δράσουν επειδή πρέπει να περιμένουν για να δουν τον καρπό των αποφάσεών τους. Λειτουργούν σε περιβάλλον καθυστερημένης επιστροφής, όπου μπορεί να είναι δύσκολο να κατανοήσουμε ποια ενέργεια οδηγεί σε ποιο αποτέλεσμα σε πολλά χρονικά βήματα. Οι αλγόριθμοι ενισχυτικής μάθησης αποδίδουν σιγά-σιγά όλο και καλύτερα σε πιο διφορούμενα περιβάλλοντα της πραγματικής ζωής, ενώ επιλέγουν από έναν αυθαίρετο αριθμό πιθανών ενεργειών. Δηλαδή, αρχίζουν να πετυχαίνουν στόχους στον πραγματικό κόσμο χάρης την παρέμβασε της βαθειάς ενισχυτικής μάθησης.[53]

Ευτυχώς, υπάρχουν πολλές πραγματικές εφαρμογές του DRL. Ένα από τα γνωστά είναι στον τομέα των αυτοκινήτων χωρίς οδηγό. Στην κατασκευή, τα ευφυή ρομπότ εκπαιδεύονται χρησιμοποιώντας DRL για να τοποθετούν αντικείμενα στη σωστή θέση, μειώνοντας το κόστος εργασίας και αυξάνοντας την παραγωγικότητα. Στις σημερινές επιχειρηματικές δραστηριότητες, το DRL χρησιμοποιείται εκτενώς στη διαχείριση της εφοδιαστικής αλυσίδας, στην πρόβλεψη ζήτησης, στη διαχείριση αποθεμάτων, στη διαχείριση των εργασιών αποθήκης κ.λπ. Το DRL έχει χρησιμοποιηθεί συνήθως σε πολλές εργασίες Επεξεργασίας Φυσικής Γλώσσας (NLP), όπως περίληψη αφηρημένης κειμένου και chatbots. Πολλές πρόσφατες ερευνητικές εργασίες προτείνουν εφαρμογές του DRL στην υγειονομική περίθαλψη, τα εκπαιδευτικά συστήματα, τις έξυπνες πόλεις, μεταξύ πολλών άλλων. Συνοπτικά, κανένας επιχειρηματικός τομέας δεν μένει ανεπηρέαστος από το DRL.[54]

Οι πράκτορες DRL μπορούν μερικές φορές να ελέγχουν επικίνδυνα περιβάλλοντα της πραγματικής ζωής, όπως ρομπότ ή αυτοκίνητα, γεγονός που αυξάνει τον κίνδυνο λανθασμένων επιλογών. Υπάρχει ένα σημαντικό πεδίο που ονομάζεται ασφαλής RL που προσπαθεί να αντιμετωπίσει αυτόν τον κίνδυνο, για παράδειγμα, μαθαίνοντας μια πολιτική που μεγιστοποιεί τις ανταμοιβές ενώ λειτουργεί εντός προκαθορισμένων περιορισμών ασφαλείας. Επίσης, οι πράκτορες DRL κινδυνεύουν επίσης από μια επίθεση, όπως κάθε άλλο σύστημα λογισμικού. Αλλά το DRL προσθέτει μερικά νέα διανύσματα επίθεσης πέρα από τα παραδοσιακά συστήματα μηχανικής μάθησης, επειδή, γενικά, έχουμε να κάνουμε με συστήματα πολύ πιο περίπλοκα στην κατανόηση και τη μοντελοποίηση.[54]

5.1 Critic methods - Q-learning

Δυστυχώς, το Q-learning από μόνο του δεν καλύπτει όλες τις ανάγκες που θα θέλαμε. Ειδικότερα, παρέχει κακή απόδοση σε περίπτωση που το μοντέλο έχει να εκτελέσει πολλές ενέργειες. Η διερεύνηση όλων των πιθανών ενεργειών χρησιμοποιώντας μια ε-greedyστρατηγική μπορεί να διαρκέσει πολύ και ο αλγόριθμος μπορεί εύκολα να συγκλίνει σε τοπικά μέγιστα παρά στην πραγματική καλύτερη δυνατή λύση. Επιπλέον, το σύνολο των ενεργειών πρέπει να είναι πεπερασμένο, δηλαδή εάν οι ενέργειες που πρέπει να εκτελέσει το μοντέλο είναι άπειρες, αν και ασυνήθιστο, ο αλγόριθμος Q-learning δεν θα μπορέσει να ανταποκριθεί. Τέλος, θεωρείται πως ο αλγόριθμος αυτός παρέχει κακή στρατηγική

εξερεύνησης, διότι εάν για παράδειγμα ο πράκτορας βρίσκεται σε ένα περιβάλλον, πρώτα θα εκτελέσει την ενέργεια με την μεγαλύτερη πιθανότητα. Αυτή συνήθως είναι η πρώτη ενέργεια, η οποία έχει πιθανότητα 91%, 90 από την ενέργεια greedy και 1% από τις πιθανότητες μα επιλεχτεί κάποια ενέργεια. Αυτό σημαίνει πως η πρώτη κίνηση του πράκτορα ίσως να μην είναι και η καλύτερη κίνηση.

Ένας αλγόριθμος Q-Learning μαθαίνει προσπαθώντας να βρει τη συνάρτηση τιμής δράσης κάθε κατάστασης — τη συνάρτηση Q-Value. Ολόκληρη η διαδικασία εκμάθησής του βασίζεται στην ιδέα της εκτίμησης της ποιότητας κάθε πιθανής ενέργειας και της επιλογής σύμφωνα με αυτή τη γνώση. Έτσι, το Q-Learning προσπαθεί να έχει πλήρη και αμερόληπτη γνώση όλων των πιθανών κινήσεων, το οποίο είναι και το σημαντικότερο μειονέκτημά του, καθώς αυτό απαιτεί επαρκή αριθμό προσπαθειών για κάθε πιθανή κατάσταση και ενέργεια. Ο αλγόριθμος Policy Gradient μαθαίνει με πιο εύρωστο τρόπο, όχι προσπαθώντας να αξιολογήσει την αξία κάθε ενέργειας αλλά απλώς αξιολογώντας ποια ενέργεια θα προτιμούσε. Γενικά, κάθε μοντέλο έχει τελικά κάποιες παραμέτρους που μάθησαν τις οποίες προσπαθεί να βελτιστοποιήσει (αυτές είναι συνήθως τα βάρη του νευρωνικού δικτύου). Ας χαρακτηρίσουμε αυτές τις παραμέτρους ως θ. Στο DQN, ο κύκλος μάθησης είναι:

$$\text{Training} \rightarrow \theta \rightarrow \text{Q-Value} \rightarrow \pi$$

Εικόνα 55: Κύκλος Μάθησης DQN

Αυτό σημαίνει ότι η διαδικασία εκπαίδευσης βελτιστοποιεί τις μαθημένες παραμέτρους του δικτύου, οι οποίες στη συνέχεια χρησιμοποιούνται για τον υπολογισμό των Q-Values. Στη συνέχεια χρησιμοποιούμε αυτές τις τιμές Q για να αποφασίσουμε για την πολιτική μας π, όπου μια πολιτική είναι απλώς η πιθανότητα επιλογής κάθε ενέργειας. Η πολιτική ε-άπληστη του DQN εξαρτάται από τις προβλεπόμενες τιμές Q, καθώς δίνει την υψηλότερη πιθανότητα στην ενέργεια με η υψηλότερη τιμή Q:

$$\pi_{dqn}(a) = \begin{cases} \underset{a}{\text{argmax}}\, Q(s,a) & r > \epsilon \\ \text{random action} & r < \epsilon \end{cases} \quad \text{for } r \sim U(0,1)$$

Εικόνα 56: Εξαρτώμενες μεταβλητές της πολιτικής ε-άπληστης του DQN

Ο κύκλος εκμάθησης ενός αλγόριθμου Gradient πολιτικής είναι πιο σύντομος, παρακάμπτοντας το τμήμα Q-Value:

$$\text{Training} \rightarrow \theta \rightarrow \pi$$

Εικόνα 57: Ο κύκλος εκμάθησης ενός αλγόριθμου Gradient

Αυτό σημαίνει ότι το δίκτυο θα δώσει απευθείας έξοδο την πιθανότητα επιλογής κάθε ενέργειας, παρακάμπτοντας τον επιπλέον υπολογισμό. Αυτό θα επιτρέψει σε αυτόν τον αλγόριθμο να είναι πιο ισχυρός.

Επιπλέον, ο κύριος στόχος για τη βελτιστοποίηση των βαρών ενός δικτύου στο DQN είναι να διασφαλίσουμε ότι το μοντέλο βελτιώνεται μετά από κάθε φάση εκμάθησης. Ας προσπαθήσουμε να το γράψουμε ως έναν πολύ γενικό κανόνα ενημέρωσης των βαρών σε κάθε χρονικό βήμα t ως:

$$\theta_{t+1} = \theta_t + a\nabla J(\theta_t)$$

Όπου ορίσαμε το J ως κάποιο μέτρο απόδοσης, στο οποίο θα φτάσουμε σε ένα δευτερόλεπτο, και το α είναι ένας ρυθμός μάθησης. Οι καταστάσεις ορίζονται από το περιβάλλον καθώς και τα βήματα ορίζονται από τον πράκτορα και επισημαίνουν την ακολουθία των καταστάσεων που έχει περάσει. Αυτό σημαίνει ότι το βήμα t=3 και t=7 μπορεί να είναι η ίδια κατάσταση s, καθώς ορισμένα περιβάλλοντα επιτρέπουν σε έναν πράκτορα να επιστρέψει στην ίδια ακριβώς κατάσταση πολλές φορές. Από την άλλη πλευρά, κάθε βήμα t συμβαίνει μία και μόνο μία φορά σε κάθε επεισόδιο. Τα βήματα είναι μια χρονολογική σειρά χρόνου.

Όπως προαναφέρθηκε, στόχος είναι να βελτιωθεί η απόδοση, επομένως θέλουμε να μεγιστοποιήσουμε την παράγωγο του J και όχι να την ελαχιστοποιήσουμε. Αυτό είναι γνωστό ως ανάβαση κλίσης — σε αντίθεση με την κλίση κατάβασης που πραγματοποιείται κατά την οπισθοδιάδοση (back-propagation).

Σε αυτό το σημείο είναι προφανές ότι ο τρόπος με τον οποίο ορίζουμε το J θα κάνει τη διαφορά, αλλά είναι επίσης αποκλειστικά στο χέρι μας να αποφασίσουμε ποιος είναι ο σωστός ορισμός. Έστω ότι θέλουμε να εντοπίσουμε κάτι απλό που εξαρτάται από την Q-Value. Αρκεί να θυμηθούμε ότι μια Q-Value Q(s,a) είναι ένα μέτρο όλων των ανταμοιβών που θα λάβει ο πράκτορας μέχρι το τέλος του επεισοδίου, ξεκινώντας από την κατάσταση s και μετά την εκτέλεση της ενέργειας a. Έτσι, το άθροισμα όλων των ανταμοιβών φαίνεται να είναι μια αρκετά καλή μέτρηση απόδοσης, σε τελική ανάλυση, ο μοναδικός σκοπός του πράκτορα είναι να αυξήσει τις συνολικές ανταμοιβές που συλλέγονται. Έτσι, μπορούμε να χρησιμοποιήσουμε ως μέτρο απόδοσης τις συνολικές συσσωρευμένες ανταμοιβές σε ένα ολόκληρο επεισόδιο:

$$J = [\sum_{a} Q(S_{0,}\alpha) \cdot p(a)]$$

Αυτό είναι το άθροισμα όλων των Q-Τιμών της αρχικής κατάστασης s^o, πολλαπλασιαζόμενο με την πιθανότητα για την επιλογή κάθε ενέργειας. Αυτή ακριβώς είναι η αναμενόμενη συνολική μας ανταμοιβή ενός ολόκληρου επεισοδίου. Τώρα, αυτή η άθροιση όλων των Q-Τιμών μιας συγκεκριμένης κατάστασης πολλαπλασιαζόμενη με τις πιθανότητές τους έχει ένα όνομα: ονομάζεται συνάρτηση τιμής κατάστασης και συμβολίζεται ως V(s). Η συνάρτηση αξίας μιας πολιτείας είναι ένα μέτρο της αναμενόμενης ανταμοιβής από την κατάσταση και μέχρι το τέλος του επεισοδίου, χωρίς να γνωρίζουμε ποια ενέργεια θα επιλεγεί. Οπότε ορίσαμε βασικά τη συνάρτηση τιμής της αρχικής κατάστασης ως μέτρο απόδοσης: J = $V(s^o)$.

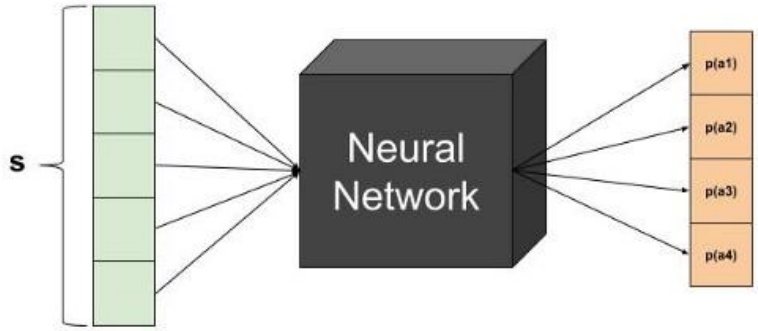
Τώρα που έχουμε το J, πρέπει να υπολογίσουμε την κλίση του. Αυτό περιλαμβάνει κάποια πολύπλοκα μαθηματικά, καθώς το J εξαρτάται από την πιθανότητα επιλογής ενέργειας p(a), η οποία προέρχεται από την πολιτική π αλλά το π εξαρτάται από το J, αφού έτσι την ορίσαμε.

$$\nabla J(\theta_t) = G_t \nabla ln\pi(\alpha_{t|} s_{t,} \theta_t)$$

Όπου G_t είναι η συνολική συσσωρευμένη ανταμοιβή από το βήμα t μέχρι το τέλος του επεισοδίου. Αυτό σημαίνει ότι ο κανόνας ενημέρωσης για το θ μας είναι:

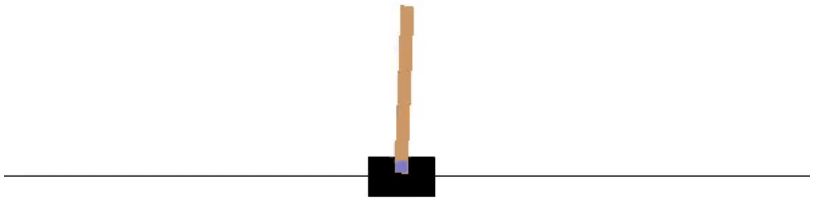
$$\theta_{t+1} = \theta_t + a\big(G_t \cdot \nabla_\theta \, ln\pi(\alpha_t|S_t, \theta_t)\big)$$

Επιπροσθέτως, σημαντικό χαρακτηριστικό του αλγορίθμου Policy Gradient είναι ότι μαθαίνει απευθείας την πολιτική. Αυτό σημαίνει ότι η είσοδος του δικτύου είναι η τρέχουσα κατάσταση s και η έξοδος είναι η πιθανότητα επιλογής κάθε ενέργειας:



Εικόνα 58: Policy Gradient – Είσοδος-Έξοδος

Άρα, το μοντέλο επιλέγει την ενέργεια που πρέπει να κάνει, εκτελώντας μια σταθμισμένη δειγματοληψία πάνω από την ενέργεια. Αυτό λύνει επίσης την ανάγκη μας για εξερεύνηση — κάθε ενέργεια έχει πιθανότητες να επιλεγεί, αλλά όχι εξίσου, καθώς η καλύτερη ενέργεια θα είναι η πιο πιθανό να επιλεγεί. Σε αντίθεση με το Q-Learning, ο αλγόριθμος Policy Gradient είναι ένας αλγόριθμος εντός πολιτικής, πράγμα που σημαίνει ότι μαθαίνει μόνο χρησιμοποιώντας μεταβάσεις κατάστασης-ενέργειας που πραγματοποιούνται από την τρέχουσα ενεργή πολιτική. Τεχνικά, αυτό σημαίνει ότι δεν υπάρχει μνήμη Επανάληψης Εμπειρίας όπως στο DQN. Μόλις το μοντέλο εκπαιδευτεί, οι παράμετροι θ του αλλάζουν και επομένως και η πολιτική του. Αυτό σημαίνει ότι όλη η εμπειρία που συλλέγεται πριν από αυτήν την εκπαίδευση πρέπει να απορριφθεί και δεν μπορεί πλέον να χρησιμοποιηθεί για εκπαίδευση. Έτσι, κάθε κομμάτι δεδομένων που συλλέγουμε για εκπαίδευση χρησιμοποιείται μία και μόνο μία φορά. Ενώ ελπίζουμε ότι η γενική ιδέα είναι κατανοητή σε αυτό το σημείο, ένα παράδειγμα είναι πάντα καλύτερο. Μια πολύ δημοφιλής εργασία που πρέπει να λυθεί χρησιμοποιώντας την Ενισχυτική Μάθηση είναι το πρόβλημα Cart-Pole: ένα καρότσι, που μπορεί να κινηθεί είτε αριστερά είτε δεξιά, πρέπει να βεβαιωθεί ότι ο πόλος που βρίσκεται πάνω του δεν πέφτει. Εδώ είναι η λύση μου σε αυτήν την πρόκληση.

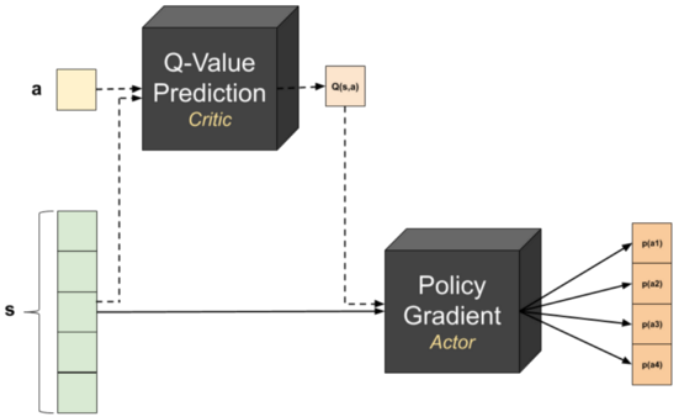


Εικόνα 59: Πρόβλημα Cart-Pole

Συνοψίζοντας, μέχρι τώρα αναφέρθηκε πως ένας πράκτορας μαθαίνει μια πολιτική χωρίς να χρειάζεται να μάθει την πραγματική αξία κάθε ενέργειας. Το μόνο πράγμα που χρειάζεται πραγματικά είναι κάποια μέτρηση απόδοσης, την οποία θα προσπαθήσει να μεγιστοποιήσει και στην περίπτωσή μας επιλέξαμε τη συνολική αναμενόμενη ανταμοιβή ενός ολόκληρου επεισοδίου. Τα οφέλη αυτής της μεθόδου είναι ξεκάθαρα ότι δεν χρειάζεται η δοκιμή κάθε πιθανού ζεύγους κατάστασης-δράσεων, καθώς ο Πράκτορας αναπτύσσει κάποιο είδος «εντερικής αίσθησης» σχετικά με το τι πρέπει να κάνει. Αλλά αυτό έρχεται σε σύγκρουση με ένα μειονέκτημα λόγω της απόλυτης εξάρτησης του πράκτορα στη μέτρηση απόδοσής του.

Ωστόσο, αναπτύχθηκε μια καινούρια μέθοδος, γνωστή ως Actor-Critic, η οποία αποτελείται από δύο δευτερεύοντες πράκτορες που μαθαίνουν μαζί, όπου ο ένας μαθαίνει την πολιτική που πρέπει να ενεργεί και επομένως είναι γνωστή ως Actor και ο άλλος μαθαίνει την Q-Value του καθενός κατάσταση και δράση και επομένως είναι γνωστός ως Critic. Στη συνέχεια αλλάζουμε τον κανόνα ενημέρωσης του θ μας ως εξής:

$$\theta_{t+1} = \theta_t + a\big(Q(S_t, \alpha_t) \cdot \nabla_\theta \, ln\pi(\alpha_t|S_t, \theta_t)\big)$$



Εικόνα 60: Μια σχηματική άποψη μιας αρχιτεκτονικής ηθοποιού-κριτικού

Στην εικόνα 60, οι διακεκομμένες γραμμές αντιπροσωπεύουν ροές που αφορούν μόνο κατά τη διάρκεια της εκπαίδευσης, επομένως κατά τη φάση εξαγωγής συμπερασμάτων (πρόβλεψης), χρησιμοποιείται μόνο ο Actor. Παρατηρώντας τη φωτογραφία, βλέπουμε την αντικατάσταση της συνολικής ανταμοιβής G με την Q-Value αλλά η Q-Value είναι πλέον επίσης μια μαθημένη παράμετρος που μαθαίνεται από τον δεύτερο υπο-πράκτορα. Αυτό αποδίδει λίγο πιο περίπλοκη αρχιτεκτονική.[55]

5.2 Actor methods - Policy gradient

Στο RL που βασίζεται σε πολιτικές, η βέλτιστη πολιτική υπολογίζεται με απευθείας χειρισμό της πολιτικής και η συνάρτηση βάσει αξίας βρίσκει σιωπηρά τη βέλτιστη πολιτική βρίσκοντας τη συνάρτηση βέλτιστης τιμής. Το RL που βασίζεται σε πολιτικές είναι αποτελεσματικό σε χώρους συνεχούς δράσης υψηλών διαστάσεων και στοχαστικών και εκμάθησης στοχαστικών πολιτικών. Ταυτόχρονα, το RL με βάση την αξία υπερέχει στην αποτελεσματικότητα και τη σταθερότητα του δείγματος. Οι συναρτήσεις που βασίζονται σε τιμές προσδιορίζουν έμμεσα τη βέλτιστη πολιτική προσδιορίζοντας τη συνάρτηση ιδανικής τιμής, αλλά η RL που βασίζεται σε πολιτική χειρίζεται ρητά την πολιτική για να καθορίσει την καλύτερη πολιτική. Σε χώρους συνεχούς δράσης υψηλών διαστάσεων και στοχαστικών, καθώς και κατά την εκμάθηση στοχαστικών πολιτικών, το RL που βασίζεται σε πολιτικές είναι επιτυχές. Το RL με βάση τις τιμές υπερέχει ως προς την αποτελεσματικότητα και τη σταθερότητα δειγματοληψίας ταυτόχρονα.

Η σημαντική διακύμανση της κλίσης είναι η κύρια δυσκολία με την κλίση πολιτικής RL. Η χρήση της βασικής συνάρτησης b(st) για τη μείωση της διακύμανσης στην εκτίμηση της κλίσης είναι η συμβατική στρατηγική. Υπάρχουν πολλές ανησυχίες που περιλαμβάνουν τη γραμμή βάσης που μπορεί να εισάγουν μεροληψία στην εκτίμηση της κλίσης. Υπάρχουν ενδείξεις ότι η βασική γραμμή δεν παρέχει την εκτίμηση της κλίσης με βάση.

Ο τύπος του Policy gradient είναι ο εξής:

$$\nabla J(\theta) = \nabla \varepsilon_{\pi\theta} |R(T)| = \varepsilon_{\pi\theta} (\sum_{t=0}^{T=1} \nabla_\theta \, log_{\Pi_\theta} (\alpha_\theta \mid s_\theta) R(T)$$

Όπου ο τύπος της ανταμοιβής, R(τ), είναι:

$$R(T) = \sum_{\tau=0}^{-1} r_t$$

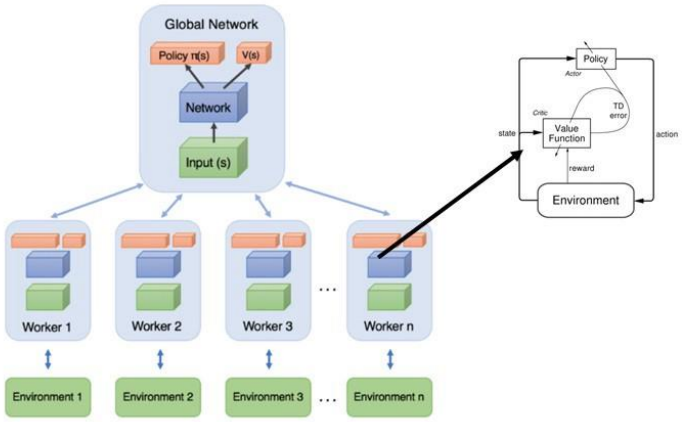
Με έναν απλό όρο, το Actor-Critic είναι μια έκδοση Temporal Difference (TD) της Policy gradient. Έχει δύο δίκτυα, τον Ηθοποιό και τον Κριτικό. Ο ηθοποιός αποφασίζει ποια δράση πρέπει να γίνει και ο κριτικός ενημερώνει τον ηθοποιό πόσο καλή ήταν η δράση και πώς έπρεπε να προσαρμοστεί. Η μάθηση του παράγοντα βασίζεται στην προσέγγιση της κλίσης πολιτικής. Συγκριτικά, οι κριτικοί αξιολογούν τη δράση που παράγεται από τον ηθοποιό υπολογίζοντας τη συνάρτηση τιμής. Αυτός ο τύπος αρχιτεκτονικής βρίσκεται στο Generative Adversarial Network (GAN) όπου τόσο ο διαχωριστής όσο και ο δημιουργός συμμετέχουν σε ένα παιχνίδι. Η γεννήτρια δημιουργεί τις ψεύτικες εικόνες και ο διαχωριστής αξιολογεί πόσο καλή είναι η ψεύτικη εικόνα που δημιουργείται με την αναπαράσταση της πραγματικής εικόνας. Με την πάροδο του χρόνου, το Generator μπορεί να δημιουργήσει ψεύτικες εικόνες που δεν μπορούν να διακριθούν από τον χρήστη. Ομοίως, ο Actor και ο Critic συμμετέχουν στο παιχνίδι, αλλά και οι δύο βελτιώνονται με την πάροδο του χρόνου, σε αντίθεση με το GAN. Το Actor-critic είναι παρόμοιο με έναν αλγόριθμο διαβάθμισης πολιτικής που ονομάζεται REINFORCE with

baseline. Ενίσχυση είναι η εκμάθηση MONTE-CARLO που δείχνει ότι η συνολική απόδοση λαμβάνεται από την πλήρη τροχιά. Αλλά στον ηθοποιό-κριτικό, χρησιμοποιούμε bootstrap. Έτσι οι κύριες αλλαγές στη συνάρτηση πλεονεκτήματος.

5.3 Actor-Critic methods - A2C, A3C

Χρησιμοποιώντας το Monte Carlo, δημιούργουνται κάποια προβλήματα. Ένα από αυτά είναι η υψηλή μεταβλητότητα στις πιθανότητες καταγραφής (καταγραφή της κατανομής πολιτικής) και σωρευτικές τιμές ανταμοιβής, επειδή κάθε τροχιά κατά τη διάρκεια της εκπαίδευσης μπορεί να αποκλίνει η μία από την άλλη σε μεγάλο βαθμό. Κατά συνέπεια, η υψηλή μεταβλητότητα στις πιθανότητες καταγραφής και τις σωρευτικές τιμές ανταμοιβής θα δημιουργήσει θορυβώδεις κλίσεις και θα προκαλέσει ασταθή μάθηση ή/και την λοξή κατανομή της πολιτικής προς μια μη βέλτιστη κατεύθυνση. Εκτός από την υψηλή διακύμανση των κλίσεων, ένα άλλο πρόβλημα με τις κλίσεις πολιτικής εμφανίζεται ότι οι τροχιές έχουν σωρευτική ανταμοιβή 0. Η ουσία της κλίσης πολιτικής είναι η αύξηση των πιθανοτήτων για «καλές» ενέργειες και η μείωση αυτών των «κακών» ενεργειών στην κατανομή της πολιτικής. Τόσο οι "καλές" τόσο οι "κακές" ενέργειες δεν θα μαθευτούν εάν η αθροιστική ανταμοιβή είναι 0. Τέλος, αυτά τα ζητήματα συμβάλλουν στην αστάθεια και την αργή σύγκλιση των μεθόδων κλίσης της πολιτικής βανίλιας (vanilla policy gradient).

Το πλεονέκτημα του Actor Critic είναι ότι έχει δύο κύριες παραλλαγές: το Asynchronous Advantage Actor Critic (A3C) και το Advantage Actor Critic (A2C). Το A3C εισήχθη στην εργασία του Deepmind «Asynchronous Methods for Deep Reinforcement Learning» (Mnih et al, 2016). Ουσιαστικά, το A3C εφαρμόζει παράλληλη εκπαίδευση όπου πολλοί εργαζόμενοι σε παράλληλα περιβάλλοντα ενημερώνουν ανεξάρτητα μια συνάρτηση συνολικής αξίας - επομένως "ασύγχρονη". Ένα βασικό πλεονέκτημα της ύπαρξης ασύγχρονων δρώντων είναι η αποτελεσματική και αποδοτική εξερεύνηση του κρατικού χώρου.



Εικόνα 61: Αρχιτεκτονική υψηλού επιπέδου του A3C

Το A2C είναι σαν το A3C αλλά χωρίς το ασύγχρονο μέρος. αυτό σημαίνει μια παραλλαγή του A3C για ένα άτομο. Βρέθηκε εμπειρικά ότι το A2C παράγει συγκρίσιμες επιδόσεις με το A3C ενώ είναι πιο αποτελεσματικό. Σύμφωνα με αυτήν την ανάρτηση ιστολογίου OpenAI, οι ερευνητές δεν είναι απολύτως σίγουροι εάν ή πώς ο ασυγχρονισμός ωφελεί τη μάθηση.

Η σύγχρονη εφαρμογή μας A2C αποδίδει καλύτερα από τις ασύγχρονες υλοποιήσεις μας — δεν έχουμε δει καμία ένδειξη ότι ο θόρυβος που εισάγεται από τον ασύγχρονο παρέχει κάποιο όφελος απόδοσης. Αυτή η υλοποίηση A2C είναι πιο οικονομική από την A3C όταν χρησιμοποιούνται μηχανές με μία GPU και είναι ταχύτερη από μια εφαρμογή A3C μόνο με CPU όταν χρησιμοποιούνται μεγαλύτερες πολιτικές.[59]

5.4 Deep Q-Network

Το DQN πρωτοπαρουσιάστηκε σε 2 σημεία, η πρώτη ήταν παίζοντας Atari με Deep Reinforcement Learning στο NIPS το 2013 και η άλλη ελέγχοντας σε ανθρώπινο επίπεδο μέσω μάθησης βαθιάς ενίσχυσης στο Nature το 2015. Είναι ενδιαφέρον ότι υπήρχαν μόνο λίγες εργασίες σχετικά με το DRN μεταξύ 2013 και 2015. Υποθέτω ότι ο λόγος ήταν ότι οι άνθρωποι δεν μπορούσαν να αναπαράγουν την υλοποίηση DQN χωρίς πληροφορίες στην έκδοση Nature.

Εικόνα 62: Πράκτορας DQN παίζει Breakout

Το DQN έχει 4 τεχνικές που ακολουθεί. Η πρώτη αφορά την εμπειρία επανάληψης, η οποία προτάθηκε αρχικά στο Reinforcement Learning για ρομπότ που χρησιμοποιούσαν νευρωνικά δίκτυα το 1993. Το DNN υπερπροσαρμόζει εύκολα τα τρέχοντα επεισόδια. Από τη στιγμή που το DNN είναι υπερβολικά τοποθετημένο, είναι δύσκολο να παραχθούν διάφορες εμπειρίες. Για την επίλυση αυτού του προβλήματος, το Experience Replay αποθηκεύει εμπειρίες, συμπεριλαμβανομένων μεταβάσεων καταστάσεων, ανταμοιβών και ενεργειών, που είναι απαραίτητα δεδομένα για την εκτέλεση εκμάθησης Q, και δημιουργεί μικρές παρτίδες για την ενημέρωση των νευρωνικών δικτύων. Αυτή η τεχνική έχει ως πλεονέκτημα τη μείωση της συσχέτισης μεταξύ των εμπειριών στην ενημέρωση του DNN, την αύξηση της ταχύτητας εκμάθησης σε μικρές παρτίδες και την επαναχρησιμοποίηση προηγούμενων μεταβάσεων για να μην ξεχνάει το μοντέλο πράγματα. Η επόμενη τεχνική αφορά το δίκτυο-στόχο (Target Network), όπου στον υπολογισμό σφαλμάτων TD, η συνάρτηση στόχος αλλάζει συχνά με το DNN. Η ασταθής λειτουργία στόχου κάνει την προπόνηση δύσκολη. Έτσι, η τεχνική του Δικτύου Στόχου διορθώνει τις παραμέτρους της συνάρτησης στόχου και τις αντικαθιστά με το πιο πρόσφατο δίκτυο κάθε χιλιάδες βήματα.

$$Q(s_t, a) \leftarrow Q(s_t, a) + \alpha \left[r_{t+1} + \gamma \max_p Q(s_{t+1}, p) - Q(s_t, a) \right]$$

Εικόνα 63: Η συνάρτηση στόχου Q στο κόκκινο ορθογώνιο είναι σταθερή

Ακόμη μια τεχνική του DQN αφορά τις ανταμοιβές αποκοπής, όπου κάθε παιχνίδι έχει διαφορετικές κλίμακες βαθμολογίας. Για παράδειγμα, στο Pong, οι παίκτες μπορούν να πάρουν 1 πόντο όταν κερδίζουν το παιχνίδι. Διαφορετικά, οι παίκτες παίρνουν -1 βαθμό. Ωστόσο, στο Space Invaders, οι παίκτες παίρνουν 10~30 πόντους όταν νικούν εισβολείς. Αυτή η διαφορά θα έκανε την προπόνηση ασταθή. Έτσι, η τεχνική Clipping Rewards αποσπά βαθμολογίες, οι οποίες όλες οι θετικές ανταμοιβές ορίζονται +1 και όλες οι αρνητικές ανταμοιβές ορίζονται -1. Τέλος, η τελευταία τεχνική είναι η παράκαμψη πλαισίων, στην οποία το ALE είναι ικανό να αποδίδει 60 εικόνες ανά δευτερόλεπτο. Αλλά στην πραγματικότητα οι άνθρωποι δεν αναλαμβάνουν τόσο πολλές ενέργειες σε ένα δευτερόλεπτο. Το AI δεν χρειάζεται να υπολογίζει τις τιμές Q σε κάθε πλαίσιο. Έτσι, η τεχνική Skipping Frames είναι ότι το DQN υπολογίζει τις τιμές Q κάθε 4 καρέ και χρησιμοποιεί τα τελευταία 4 καρέ ως εισόδους. Αυτό μειώνει το υπολογιστικό κόστος και συγκεντρώνει περισσότερες εμπειρίες.

Όλες οι παραπάνω τεχνικές επιτρέπουν στο DQN να επιτύχει σταθερή εκπαίδευση.

	Breakout	R. Raid	Enduro	Sequest	S. Invaders
DQN	**316.8**	**7446.6**	**1006.3**	**2894.4**	**1088.9**
Naive DQN	3.2	1453.0	29.1	275.8	302.0
Linear	3.0	2346.9	62.0	656.9	301.3

Εικόνα 64: Το DQN κατακλύζει το αφελές DQN

Στην έκδοση Nature, δείχνει πόσο το Experience Replay και το Target Network συμβάλλουν στη σταθερότητα.

Replay	○	○	×	×
Target	○	×	○	×
Breakout	**316.8**	240.7	10.2	3.2
River Raid	**7446.6**	4102.8	2867.7	1453.0
Seaquest	**2894.4**	822.6	1003.0	275.8
Space Invaders	**1088.9**	826.3	373.2	302.0

Εικόνα 65: Απόδοση με και χωρίς Experience Replay και Target Network

Συμπερασματικά, βλέπουμε πως το Experience Replay είναι πολύ σημαντικό στο DQN και το Target Network αυξάνει επίσης την απόδοσή του.[62]

5.5 PPO

Η διαδρομή προς την επιτυχία στην ενισχυτική μάθηση δεν είναι τόσο προφανής - οι αλγόριθμοι έχουν πολλά κινούμενα μέρη που είναι δύσκολο να εντοπιστούν σφαλμάτων και απαιτούν σημαντική προσπάθεια συντονισμού προκειμένου να ληφθούν καλά αποτελέσματα. Το PPO επιτυγχάνει μια ισορροπία μεταξύ της ευκολίας υλοποίησης, της πολυπλοκότητας του δείγματος και της ευκολίας συντονισμού, προσπαθώντας να υπολογίσει μια ενημέρωση σε κάθε βήμα που ελαχιστοποιεί τη συνάρτηση κόστους ενώ διασφαλίζει ότι η απόκλιση από την προηγούμενη πολιτική είναι σχετικά μικρή.

$$L^{clip}(\theta) = \hat{\varepsilon}_t[\min(r_t(\theta)\hat{A}_{t,}\text{clip}(\boldsymbol{r_t}(\boldsymbol{\theta}), 1-\varepsilon, 1+\varepsilon)\hat{A}_{t,})]$$

Όπου θ είναι η παράμετρος πολιτικής, $\hat{\varepsilon}_t$ υποδηλώνει την εμπειρική προσδοκία σε χρονικά βήματα, r_t είναι η αναλογία της πιθανότητας σύμφωνα με τη νέα και την παλιά πολιτική, $\widehat{A}_t$ είναι το εκτιμώμενο πλεονέκτημα τη χρονική στιγμή t και το ε είναι η υπερπαράμετρος, συνήθως 0,1 ή 0,2. Σε δοκιμές, αυτός ο αλγόριθμος έχει εμφανίσει την καλύτερη απόδοση σε εργασίες συνεχούς ελέγχου και σχεδόν ταιριάζει με την απόδοση του ACER στο Atari, παρόλο που είναι πολύ πιο απλός στην εφαρμογή του.

Η κατανόηση του Proximal Policy Optimization (PPO) σχετικά με την διδασκαλία ενός πράκτορα AI θα γίνει πιο κατανοητή με το παρακάτω παράδειγμα.

Εικόνα 66: Το Google Football Environment κυκλοφόρησε για έρευνα RL

Μια τυπική ρύθμιση Ενισχυτικής Μάθησης λειτουργεί με την αλληλεπίδραση ενός πράκτορα AI με το περιβάλλον μας. Ο πράκτορας παρατηρεί την τρέχουσα κατάσταση του περιβάλλοντός μας και με βάση κάποια πολιτική παίρνει την απόφαση να λάβει μια συγκεκριμένη ενέργεια. Αυτή η ενέργεια στη συνέχεια αναμεταδίδεται πίσω στο περιβάλλον το οποίο κινείται προς τα εμπρός κατά ένα βήμα. Αυτό δημιουργεί μια ανταμοιβή που υποδεικνύει εάν η ενέργεια που έγινε ήταν θετική ή αρνητική στο πλαίσιο του παιχνιδιού που παίζεται. Χρησιμοποιώντας αυτήν την ανταμοιβή ως ανατροφοδότηση, ο πράκτορας προσπαθεί να καταλάβει πώς να τροποποιήσει την υπάρχουσα πολιτική του προκειμένου να λάβει καλύτερες ανταμοιβές στο μέλλον.

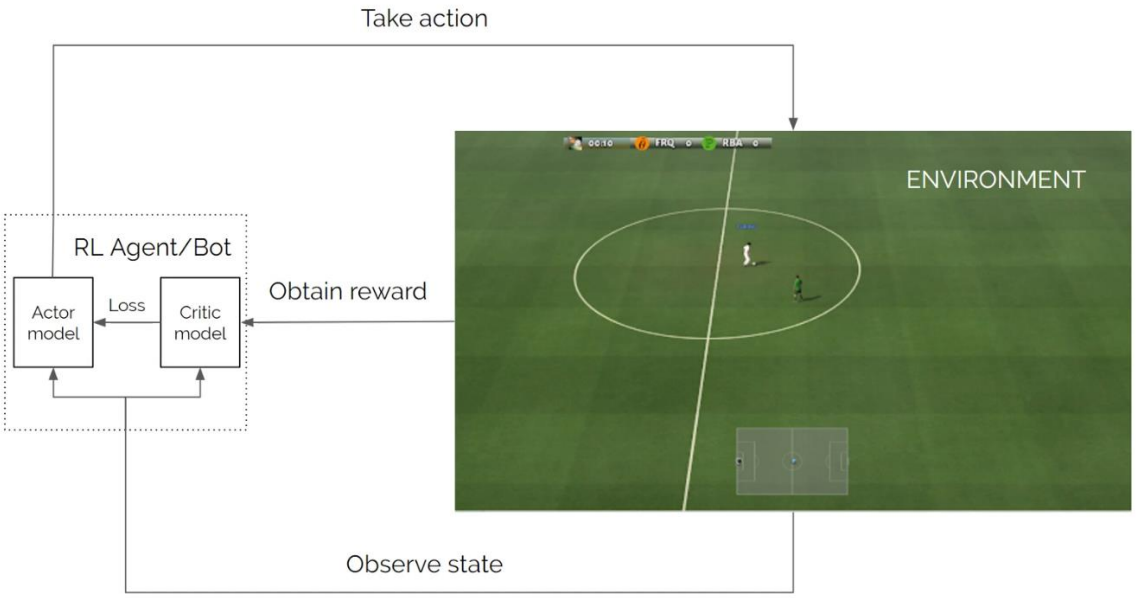
Εικόνα 67: Αλληλεπίδραση πράκτορα AI με το περιβάλλον

Αυτό δημιουργεί ένα αντικείμενο περιβάλλοντος όπου ο παίκτης βγαίνει στη μισή γραμμή και πρέπει να σκοράρει σε ένα κενό τέρμα στη δεξιά πλευρά. Αναπαράσταση 'pixels' σημαίνει ότι η κατάσταση που θα παρατηρήσει ο πράκτορας μας έχει τη μορφή εικόνας RGB του πλαισίου που αποδίδεται στην οθόνη.

Ο αλγόριθμος PPO εισήχθη από την ομάδα OpenAI το 2017 και γρήγορα έγινε μια από τις πιο δημοφιλείς μεθόδους RL που σφετερίζονται τη μέθοδο εκμάθησης Deep-Q. Περιλαμβάνει τη συλλογή μιας μικρής παρτίδας εμπειριών που αλληλοεπιδρούν με το περιβάλλον και τη χρήση αυτής της παρτίδας για την ενημέρωση της πολιτικής λήψης αποφάσεων. Μόλις η πολιτική ενημερωθεί με αυτήν την παρτίδα, οι εμπειρίες απορρίπτονται και συλλέγεται μια νεότερη παρτίδα με την πρόσφατα ενημερωμένη πολιτική. Αυτός είναι ο λόγος για τον οποίο είναι μια προσέγγιση «εκμάθησης εντός πολιτικής» όπου τα δείγματα εμπειρίας που συλλέγονται είναι χρήσιμα για την ενημέρωση της τρέχουσας πολιτικής μόνο μία φορά.

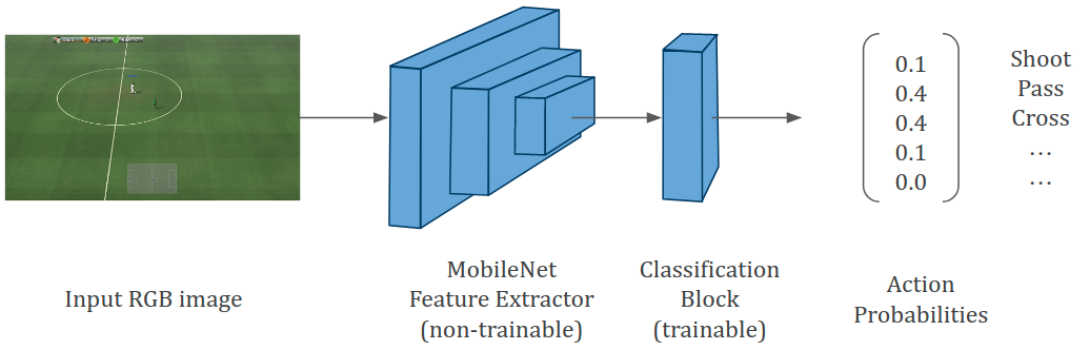
Η βασική συνεισφορά του PPO είναι να διασφαλίσει ότι μια νέα ενημέρωση της πολιτικής δεν την αλλάζει πολύ από την προηγούμενη πολιτική. Αυτό οδηγεί σε λιγότερη απόκλιση στην προπόνηση με κόστος κάποιας μεροληψίας, αλλά εξασφαλίζει πιο ομαλή εκπαίδευση και διασφαλίζει επίσης ότι ο πράκτορας δεν ακολουθεί μια ανεπανόρθωτη πορεία λήψης παράλογων ενεργειών. Λοιπόν, ας

προχωρήσουμε και ας αναλύσουμε τον πράκτορά μας AI σε περισσότερες λεπτομέρειες και ας δούμε πώς ορίζει και ενημερώνει την πολιτική του. Κατόπιν, θα χρησιμοποιήσουμε την προσέγγιση Actor-Critic για τον πράκτορά μας PPO. Χρησιμοποιεί δύο μοντέλα, και τα δύο Deep Neural Nets, το ένα ονομάζεται Actor και το άλλο που ονομάζεται Critic.



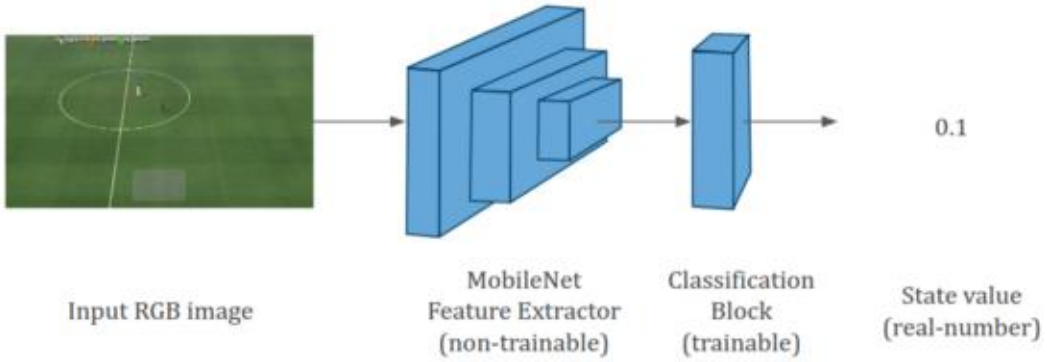
Εικόνα 68: PPO Agent

Το μοντέλο Actor εκτελεί το καθήκον της εκμάθησης της δράσης που πρέπει να γίνει κάτω από μια συγκεκριμένη παρατηρούμενη κατάσταση του περιβάλλοντος. Στην περίπτωσή μας, παίρνει την εικόνα RGB του παιχνιδιού ως είσοδο και δίνει μια συγκεκριμένη ενέργεια όπως σουτ ή πάσα ως έξοδο.



Εικόνα 69: The Actor model

Στέλνουμε τη δράση που προβλέπει ο Ηθοποιός στο ποδοσφαιρικό περιβάλλον και παρατηρούμε τι συμβαίνει στο παιχνίδι. Εάν συμβεί κάτι θετικό ως αποτέλεσμα της δράσης μας, όπως το να πετύχουμε ένα γκολ, τότε το περιβάλλον στέλνει μια θετική απάντηση με τη μορφή ανταμοιβής. Αν προκύψει αυτογκόλ λόγω της δράσης μας, τότε παίρνουμε αρνητική ανταμοιβή. Αυτή η ανταμοιβή λαμβάνεται από το μοντέλο Critic.



Εικόνα 70: Η ανταμοιβή λαμβάνεται από τον Κριτή

Η δουλειά του μοντέλου της κριτικής είναι να μάθει να αξιολογεί εάν η δράση που έκανε ο Ηθοποιός οδήγησε το περιβάλλον μας σε καλύτερη κατάσταση ή όχι και να δώσει τα σχόλιά του στον Ηθοποιό, εξ ου και το όνομά του. Εξάγει έναν πραγματικό αριθμό που υποδεικνύει μια βαθμολογία (Q-value) της ενέργειας που έγινε στην προηγούμενη κατάσταση. Συγκρίνοντας αυτήν την αξιολόγηση που λαμβάνεται από τον Κριτικό, ο Ηθοποιός μπορεί να συγκρίνει την τρέχουσα πολιτική του με μια νέα πολιτική και να αποφασίσει πώς θέλει να βελτιωθεί για να κάνει καλύτερες ενέργειες.[60]

Όπως μπορείτε να δείτε, η δομή του νευρωνικού δικτύου Critic είναι σχεδόν ίδια με του Actor. Η μόνη σημαντική διαφορά είναι ότι το τελικό στρώμα του Critic εξάγει έναν πραγματικό αριθμό. Ως εκ τούτου, η ενεργοποίηση που χρησιμοποιείται είναι tanh και όχι softmax, καθώς δεν χρειαζόμαστε κατανομή πιθανοτήτων εδώ όπως με το Actor. Τώρα, ένα σημαντικό βήμα στον αλγόριθμο PPO είναι η εκτέλεση ολόκληρου αυτού του βρόχου με τα δύο μοντέλα για έναν σταθερό αριθμό βημάτων που είναι γνωστά ως βήματα PPO. Ουσιαστικά λοιπόν, αλληλεπιδρούμε με το περιβάλλον μας για συγκεκριμένο αριθμό βημάτων και συλλέγουμε τις καταστάσεις, τις ενέργειες, τις ανταμοιβές κ.λπ. που θα χρησιμοποιήσουμε για εκπαίδευση.

Το PPO χρησιμοποιεί μια αναλογία μεταξύ της πρόσφατα ενημερωμένης πολιτικής και της παλιάς πολιτικής στο βήμα ενημέρωσης. Υπολογιστικά, είναι ευκολότερο να αναπαρασταθεί αυτό στη φόρμα καταγραφής. Χρησιμοποιώντας αυτή την αναλογία μπορούμε να αποφασίσουμε πόση αλλαγή στην πολιτική είμαστε διατεθειμένοι να ανεχτούμε. Ως εκ τούτου, χρησιμοποιούμε μια παράμετρο αποκοπής epsilon ε για να διασφαλίσουμε ότι κάνουμε μόνο το μέγιστο ε% αλλαγή στην πολιτική μας κάθε φορά. Η τιμή του epsilon προτείνεται να διατηρηθεί στο 0,2 στο χαρτί. Η κρίσιμη απώλεια δεν είναι παρά η συνήθης μέση τετραγωνική απώλεια σφάλματος με τις Επιστροφές. Μπορούμε να συνδυάσουμε τις απώλειες ηθοποιών και κριτικών αν θέλουμε να χρησιμοποιήσουμε έναν παράγοντα έκπτωσης για να

τις φέρουμε στην ίδια τάξη μεγέθους. Η προσθήκη ενός όρου εντροπίας είναι προαιρετική, αλλά ενθαρρύνει το μοντέλο ηθοποιών μας να εξερευνήσει διαφορετικές πολιτικές και ο βαθμός στον οποίο θέλουμε να πειραματιστούμε μπορεί να ελεγχθεί από μια παράμετρο beta της εντροπίας.[61]

Κεφάλαιο 6ο: Υλοποίηση Παιχνιδιών Ενισχυτικής Μάθησης

Για την υλοποίηση, δημιουργήθηκαν 2 περιβάλλοντα, ένα περιβάλλον για το παιχνίδι Sudoku και ένα για το παιχνίδι Connect4, όπως και πράκτορες με σκοπό να αλληλεπιδρούν με αυτά και να μάθουν να κερδίζουν. Για την δημιουργία των περιβαλλόντων οι βιβλιοθήκες OpenAI Gym και PettingZoo αξιοποιήθηκαν, ενώ για την δημιουργία και εκπαίδευση των πρακτόρων χρησιμοποιήθηκαν οι βιβλιοθήκες PyTorch και Stable Baselines3.

Στο υπόλοιπο του κεφαλαίου θα παρουσιαστεί το περιβάλλον Sudoku που δημιουργήθηκε με την βιβλιοθήκη OpenAI Gym, το περιβάλλον Connect4 που δημιουργήθηκε με την βιβλιοθήκη PettingZoo και η δημιουργία πρακτόρων για τα περιβάλλοντα αυτά, με την χρήση Stable Baselines3 και τριών νευρωνικών δικτύων που υλοποιήθηκαν με την βιβλιοθήκη PyTorch. Τέλος, θα αναλυθούν τα αποτελέσματα που παρουσίασαν οι πράκτορες σε κάθε παιχνίδι.

6.1 Περιβάλλον Sudoku

Το Sudoku είναι ένα κλασσικό παιχνίδι αριθμητικής λογικής, το οποίο παίζεται σε ένα πλέγμα 9*9, χωρισμένο σε 9 μικρότερα υποπλέγματα 3*3, για το οποίο πλέγμα η πλειονότητα των κελιών είναι άδεια. Σε κάθε κελί του πλέγματος ο παίκτης τοποθετεί έναν αριθμό από το 1 έως το 9, με στόχο η κάθε γραμμή, στήλη και υποπλέγμα να περιέχει μοναδικούς αριθμούς.

Στην υλοποίηση μου, το περιβάλλον ορίζεται από 3 τύπους καταστάσεων του παιχνιδιού, από την νίκη, την ήττα και την μη τελειωμένη. Η νίκη ορίζεται ως η σωστή ολοκλήρωση όλου του πλέγματος, η ήττα ως η τοποθέτηση ενός λάθους αριθμού σε κελί και η μη τελειωμένη κατάσταση ορίζεται ως η κατάσταση κατά την οποία τα υπάρχοντα κελιά είναι σωστά συμπληρωμένα αλλά κελιά του πλέγματος είναι ακόμη άδεια. Για την νίκη, η ανταμοιβή που δόθηκε ήταν 10, για την ήττα -10, ενώ για την μη τελειωμένη κατάσταση η ανταμοιβή ήταν 1. Το περιβάλλον του παιχνιδιού Sudoku δημιουργήθηκε με την βιβλιοθήκη OpenAI Gym [65], η οποία προσφέρει μια διεπαφή για την αλληλεπίδραση ενός πράκτορα με αυτό. Η βιβλιοθήκη μπορεί να χρησιμοποιηθεί από πολλών ειδών βιβλιοθήκες Βαθιάς Ενισχυτικής Μάθησης, μια από τις οποίες είναι και η Stable-Baselines3, βασισμένη σε PyTorch, η οποία και χρησιμοποιήθηκε για την δημιουργία των πρακτόρων.

Κάθε περιβάλλον OpenAI Gym αποτελείται από 3 σημαντικά αντικείμενα και 2 βασικές μεθόδους. Τα αντικείμενα αποτελούν:

- **τον χώρο επιτρεπτών πράξεων (action space),** όπου ο χώρος πράξεων ορίζεται ως χώρος τύπου OpenAI Gym, με επιτρεπτούς χώρους να περιλαμβάνουν τον συνεχή χώρο Box, τους διακριτούς Discrete και MultiDiscrete, τον δυαδικό MultiBinary, όπως και συνδυασμούς χώρων, με την χρήση σύνθετων χώρων Tuple και Dict. Στο περιβάλλον Sudoku ο χώρος αυτός ορίστηκε ως ένας διακριτός χώρος MultiDiscrete 9 * 9 * 9, με τις διαστάσεις να αντιστοιχούν στην στήλη του κελιού, την γραμμή του κελιού και την τιμή που θα τοποθετηθεί μέσα στο κελί.
- **τον χώρο παρατήρησης (observation space)**, όπου ο χώρος παρατήρησης ορίζει το σχήμα και τα όρια της κατάστασης που επιστρέφεται από το περιβάλλον. Στο περιβάλλον Sudoku ο χώρος αυτός ορίστηκε ως ένας συνεχής χώρος εικόνας, τύπου Box, με ένα κανάλι(grayscale), διαστάσεις 9*9 που επιτρέπει μόνο κανονικοποιημένες τιμές μεταξύ 0 και 1. Οι επιτρεπτοί χώροι παρατήρησης είναι ίδιοι με τους επιτρεπτούς χώρους πράξεων.
- και **την εσωτερική τωρινή κατάσταση του περιβάλλοντος**, η οποία αναπαριστά την κατάσταση του παιχνιδιού και ενημερώνεται συνεχώς. Στο περιβάλλον Sudoku η κατάσταση του παιχνιδιού μοντελοποιείται όπως ακριβώς και το παιχνίδι Sudoku, με ένα πλέγμα 9*9, όπου κάθε κελί περιέχει έναν αριθμό από 1 έως 9.

Οι 2 βασικές μέθοδοι ενός περιβάλλοντος OpenAI Gym είναι:

- η **reset**, η οποία θέτει το περιβάλλον στην αρχική κατάσταση και καλείται όταν ένα επεισόδιο τελειώσει. Μάλιστα, πέρα από την επιστροφή του περιβάλλοντος σε μια αρχική κατάσταση επιστρέφει την νέα κατάσταση του παιχνιδιού πίσω. Στο περιβάλλον Sudoku, η αρχική κατάσταση δομείται με τυχαίο τρόπο και την χρήση ενός αλγορίθμου backtracking. Ο αλγόριθμος backtracking είναι ένας απλός αλγόριθμος αναζήτησης δέντρου που σταματά όταν n λύσεις στο παιχνίδι έχουν βρεθεί. Έπειτα από την επιστροφή μιας μόνο λύσης κατά την αρχικοποίηση της αρχικής κατάστασης, n' κελιά διαγράφονται με τυχαίο τρόπο ώστε να δημιουργηθεί ένα περιβάλλον στο οποίο ο παίκτης επιτρέπεται να συμπληρώσει τα άδεια κελιά.
- και η **step**, η οποία δέχεται μια πράξη ή μία σειρά πράξεων για κάθε παίκτη στο περιβάλλον τον οποίο χειρίζεται ο πράκτορας και επιστρέφει πληροφορίες σχετικά με την νέα κατάσταση του περιβάλλοντος, την ανταμοιβή ή ανταμοιβές για τον κάθε παίκτη, εάν το επεισόδιο έχει τελειώσει, καθώς και μια σειρά επιπλέον πληροφοριών σχετικά με την νέα κατάσταση του περιβάλλοντος. Αναλόγως το περιβάλλον, οι παίκτες μπορεί να είναι ένας ή και πολλοί, με τον πράκτορα συχνά να παίζει με τον εαυτό του (self-play), χειρίζοντας όλους τους παίκτες ταυτοχρόνως. Στην περίπτωση του Sudoku, ο παίκτης είναι ένας.

```
def reset(self):
    self.state = self.initial_state.copy()
    return self.state.copy()[None]
```
Μέθοδος reset για το περιβάλλον Sudoku

```
def step(self, action: np.ndarray):
    """
    :param action: Multi-discrete action within ranges [0, 0, 0] and [8, 8,
8]
    :return: observation numpy array, reward scalar, episode done boolean,
info dict
    """

    # case: position already filled
    if not np.isnan(self.state[action[0], action[1]]):
        return self._make_image_obs(), -1, False, {'episode': None,
'is_success': False}

    # case: valid action
    self.state[action[0], action[1]] = action[2] + 1  # plus one for range
1-9

    status = self._check_status()
    state = self._make_image_obs()

    # valid action sub-cases: win, unfinished, loss
    if status == self._win:
        return state, 10, True, {'episode': None, 'is_success': True}
    elif status == self._unfinished:
        return state, 1, False, {'episode': None, 'is_success': False}
    elif status == self._loss:
        return state, -10, True, {'episode': None, 'is_success': False}
```
Μέθοδος step για το περιβάλλον Sudoku.

```
def _make_image_obs(self):
    obs = self.state.copy()[None]
    obs = np.where(np.isnan(obs), 0, obs)
    return obs / 9
```

```
def get_solutions(self,
                  state: Union[None, np.ndarray] = None,
                  n_solutions: int = 1):
    if state is None:
        state = self.state

    status = self._check_status(state)
    if status == self._win:
        return [state]
    if status == self._loss:
        return None

    # 1st nan index
    nan_rows, nan_cols = np.where(np.isnan(state))
    nan_idx = [nan_rows[0], nan_cols[0]]

    solutions = []
    values = np.arange(1, 10)
    np.random.shuffle(values)

    for value in values:
        state_branch = state.copy()
        state_branch[nan_idx[0], nan_idx[1]] = value
        solution_branches = self.get_solutions(state_branch, n_solutions -
len(solutions))
        if solution_branches is not None:
            solutions += solution_branches
        if len(solutions) >= n_solutions:
            return solutions

def _check_status(self, state=None) -> int:
    if state is None:
        state = self.state
    nan_mask = np.isnan(state)

    # non-validity = loss
    # row check
    for i in range(9):
        r = state[i][~nan_mask[i]]
        if np.unique(r).size < r.size:
            return self._loss
    # column check
    for i in range(9):
        c = state[:, i][~nan_mask[:, i]]
        if np.unique(c).size < c.size:
            return self._loss
    # sub-grid check
    for i, j in product(*[[0, 3, 6]] * 2):
        subgrid = state[i:i + 3, j:j + 3][~nan_mask[i:i + 3, j:j + 3]]
        if np.unique(subgrid).size < subgrid.size:
            return self._loss

    # validity + all-filled = win
    if not nan_mask.sum():
        return self._win

    # validity = unfinished
    return self._unfinished
```

Συμπληρωματικές μέθοδοι για το περιβάλλον Sudoku

Επιπλέον, μια μέθοδος render μπορεί να οριστεί, η οποία καλείται για την απεικόνιση της κατάστασης του παιχνιδιού κατά την διάρκεια που ο πράκτορας εκπαιδεύεται ή αξιολογείται. Για το περιβάλλον Sudoku, η μέθοδος render απεικονίζει το παιχνίδι σε ένα τερματικό με το μεγαλύτερο πλέγμα, τα υποπλέγματα και τους αριθμούς που βάζει ο παίκτης να είναι εμφανή.

```
def render(self, **kwargs):
    sys.stdout.write('\n-------------------------\n')
    for i in range(9):
        sys.stdout.write('| ')
        r = self.state[i]
        for j in range(9):
            if np.isnan(r[j]):
                sys.stdout.write(' ')
            else:
                sys.stdout.write(str(int(r[j])))
            sys.stdout.write(' ')
            if (j + 1) % 3 == 0:
                sys.stdout.write('| ')
        sys.stdout.write('\n')
        if (i + 1) % 3 == 0:
            sys.stdout.write('-------------------------\n')
```

Μέθοδος render για το περιβάλλον Sudoku.

```
-------------------------
| 5 8   |       |       |
| 1   6 | 4 3 8 |   2   |
| 9   2 |     5 |   8 1 |
-------------------------
| 7 6   | 5 9 2 | 1 3 8 |
|   5   | 1   3 | 9 4 7 |
| 3 9   |   8 4 |     5 |
-------------------------
|   2 7 |   5 1 | 4 9 6 |
| 4 1   | 8 2 6 |   5   |
| 6 3 5 |     7 |     2 |
-------------------------
```

Εικόνα 71: Αναπαράσταση της κατάστασης του περιβάλλοντος Sudoku με την μέθοδο render

Τέλος, το περιβάλλον Sudoku περαιτέρω διανυσματοποιήθηκε σε 8 αντίγραφα του περιβάλλοντος που παίζουν διαδοχικά, με σκοπό την ταχύτερη εκπαίδευση των πρακτόρων.

```
class VecSudokuEnv(DummyVecEnv):
    def __init__(self, n_envs=8):
        super(VecSudokuEnv, self).__init__([partial(SudokuEnv) for _ in
range(n_envs)])
```
Διανυσματοποίηση του περιβάλλοντος Sudoku.

6.2 Connect 4

Το Connect4 είναι ένα παιχνίδι δύο παικτών, μπλε και κόκκινου, που τοποθετούν πιόνια σε 7 στήλες, με την κάθε στήλη να έχει ύψος 6 θέσεων. Στόχος κάθε παίκτη είναι να καταφέρει να τοποθετήσει πρώτος 4 εφαπτόμενα πιόνια, οριζοντίως, καθέτως ή διαγωνίως, πριν τον άλλο παίκτη.

Λόγω της πολλαπλών-παικτών (multi-agent) φύσης του παιχνιδιού, το περιβάλλον του Connect4 δημιουργήθηκε με την βιβλιοθήκη PettingZoo, που επιτρέπει σε ένα πράκτορα να χειρίζεται πολλούς παίκτες ταυτόχρονα, καθιστώντας τον ικανό να παίζει με τον εαυτό του. Η βιβλιοθήκη παρομοιάζει την OpenAI Gym, έχοντας όμως συγκεκριμένες διαφορές:

1. **Κάθε παίκτης που χειρίζεται ο πράκτορας παίζει με την σειρά και όχι ταυτόχρονα.** Αυτή η ιδιότητα του περιβάλλοντος εισάγει τον περιορισμό ενός μερικώς παρατηρήσιμου περιβάλλοντος, αφού ο κάθε παίκτης αγνοεί το τι παίζουνε οι υπόλοιποι παίκτες κατά την διάρκεια του γύρου, μαθαίνοντας το μόνο όταν δέχεται πίσω την νέα κατάσταση του περιβάλλοντος και την ανταμοιβή του. Στην περίπτωση του Connect4 όμως αυτό είναι αναγκαίο, καθώς για να ενημερωθεί ένα παίκτης για την ήττα του και να λάβει μία ανταμοιβή, θα πρέπει αυτή η ήττα να σχετίζεται με μια πράξη, σύμφωνα με την Μαρκοβιανή Διαδικασία Απόφασης. Οπότε εάν ο παίκτης χάσει, χωρίς να μπορέσει να έχει κάνει άλλη πράξη στο ενδιάμεσο, η ήττα θα πρέπει να συσχετιστεί με την προηγούμενη του πράξη.
2. Κάθε παίκτης ενημερώνεται για την ανταμοιβή του, την κατάσταση του παιχνιδιού που προέκυψε από την πράξη του και την κατάσταση (τερματισμός) του επεισοδίου στο τέλος του γύρου. Το γεγονός αυτό συνάδει και με τα παραπάνω.

Στην βιβλιοθήκη PettingZoo, τα περιβάλλοντα μπορούν να δημιουργηθούν είτε έτοιμα παραλληλοποιημένα (parallel), με τα οποία οποιαδήποτε βιβλιοθήκη Βαθιάς Ενισχυτικής Μάθησης μπορεί απευθείας να επικοινωνήσει, είτε μερικώς έτοιμα που μπορούν να παραλληλοποιηθούν (raw).

Στην περίπτωση raw περιβαλλόντων, ο προγραμματιστής πρέπει:

- Να αρχικοποιήσει τα εξής:
  - **possible_agents**: Μία λίστα με τα ονόματα των παικτών που μπορούν να παίξουν.
  - **action_spaces**: Ένα dictionary με τα ονόματα των παικτών και τους χώρους πράξεων τους.
  - **observation_spaces**: Ένα dictionary με τα ονόματα των παικτών και τους χώρους παρατήρησης τους.
  - Ένα dictionary με όνομα **metadata με ένα κλειδί is_parallelizable και τιμή True** ώστε το περιβάλλον να μπορεί να παραλληλοποιηθεί.
  - Είναι χρήσιμο να αρχικοποιηθεί ακόμη μια μεταβλητή **agent_selection** και **_agent_selector**, με την συνάρτηση agent_selector, για την εναλλαγή μεταξύ των πρακτόρων ανά κάθε βήμα.
- Να δημιουργήσει τις εξής μεθόδους:
  - **reset**: Παρόμοια με την reset ενός OpenAI Gym, χωρίς να επιστρέφεται η νέα κατάσταση του περιβάλλοντος.
    Η reset θα πρέπει να θέτει στην αρχική κατάσταση τις μεταβλητές **agents**, **rewards**, **_cumulative_rewards**, **dones**, **infos** και **agent_selection**.

- **step**: Παρόμοια με την step ενός OpenAI Gym, χωρίς να επιστρέφει τίποτα. Η step δέχεται μία πράξη και πρέπει να:
  - Καλεί την μέθοδο **_was_step_done** όταν ένας πράκτορας έχει τελειώσει.
  - Σε κάθε βήμα:
    - Να ενημερώνει την εσωτερική κατάσταση του κάθε πράκτορα.
    - Να ενημερώνει την **agent_selection** με τον επόμενο πράκτορα.
    - Να υπολογίζει τις ανταμοιβές.
  - Όταν τελειώνει ο γύρος, να ενημερώνει τις ανταμοιβές και τις τελικές παρατηρήσεις και εάν το επεισόδιο τελείωσε.
- **observation_space** και **action_space**, οι οποίες δέχονται το όνομα ενός πράκτορα και επιστρέφουν τον αντίστοιχο χώρο παρατηρήσεων και πράξεων.
- **observe**: Επιστρέφει την παρατήρηση για τον αντίστοιχο πράκτορα, μετά το πέρας του επεισοδίου.

Στην περίπτωση του Connect4, το περιβάλλον δημιουργήθηκε ως μερικώς έτοιμο, το οποίο μετατράπηκε σε ένα αξιοποιήσιμο παραλληλοποιημένο περιβάλλον με την βοήθεια ενός wrapper.

Οι καταστάσεις του παιχνιδιού κατηγοριοποιήθηκαν σε 4 βασικές, κάθε μια από τις οποίες έλαβε την εξής ανταμοιβή:

- **Λάθος**: Συμβαίνει όταν ο πράκτορας τοποθετεί ένα πιόνι σε μια ήδη γεμάτη στήλη. Ο πράκτορας δέχεται ανταμοιβή -4.
- **Νίκη/ήττα**: Στην περίπτωση αυτή ένας από τους 2 παίκτες κερδίζει, αμείβοντας τον νικητή με 10 και τον χαμένο με -10.
- **Ισοπαλία**: Συμβαίνει όταν όλες οι θέσεις στο παιχνίδι είναι γεμάτες αλλά κανείς δεν έχει κατορθώσει την τοποθέτηση 4 εφαπτόμενων πιονιών. Και οι δύο παίκτες ανταμείβονται με 5.
- **Μη τελειωμένη**: Αυτή η κατάσταση είναι τυπική ενός παιχνιδιού που ακόμη παίζεται, χωρίς όμως να υπάρχει νίκη ή πέναλτι. Η ανταμοιβή είναι 0.

```
def reset(self, **kwargs):
    self.agents = self.possible_agents[:]
    self.rewards = {agent: 0 for agent in self.agents}
    self._cumulative_rewards = self.rewards.copy()
    self.dones = {agent: False for agent in self.agents}
    self.infos = {agent: {} for agent in self.agents}
    self.states = {agent: self.initial_state.copy() for agent in
self.agents}
    self.observations = {agent: self.initial_state.copy()[1] for agent
in self.agents}

    self._agent_selector = agent_selector(self.agents)
    self.agent_selection = self._agent_selector.next()
```

Μέθοδος reset του περιβάλλοντος Connect4.

```
def step(self, action) -> None:
    agent = self.agent_selection
    if self.dones[agent]:
        return self._was_done_step(action)

    self._cumulative_rewards[agent] = 0  # don't accumulate rewards
    self._update_state(agent, action)

    if self._agent_selector.is_last():  # when the round is over
```

```
            self._compute_rewards_dones()
            self._update_obss()

        self.agent_selection = self._agent_selector.next()
        self._accumulate_rewards()  # compute rewards
```

Μέθοδος step του περιβάλλοντος Connect4.

```
def _check_win(self, state: np.ndarray):
    # current player is 1, opponent is 2
    filled = state == 1
    for conv in self.convolutions:
        if np.any(convolve2d(filled, conv, mode="valid") / self.connect
== 1):
            return True
    return False

def _compute_rewards_dones(self):
    """
    Updates rewards and dones when the round is over
    """
    rewards = []
    dones = [False, False]
    for a in self.states:
        # invalid move - no change
        if (self.states[a][0] == self.states[a][1]).all():
            rewards.append(-4)
        # win (lose)
        elif self._check_win(self.states[a][1, 0]):
            rewards = [10, -10]
            dones = [True, True]
            if self.agent_name_mapping[a]:
                rewards.reverse()
            break
        # full state - draw
        elif (self.states[a][1] > 0).all():
            rewards = [5, 5]
            dones = [True, True]
            break
        # unfinished
        else:
            rewards.append(0)

    self.rewards[self.agents[0]], self.rewards[self.agents[1]] = rewards
    self.dones[self.agents[0]], self.dones[self.agents[1]] = dones

def _update_obss(self):
    """Updates observations"""
    self.observations = {agent: self.states[agent][1] for agent in
self.agents}
```

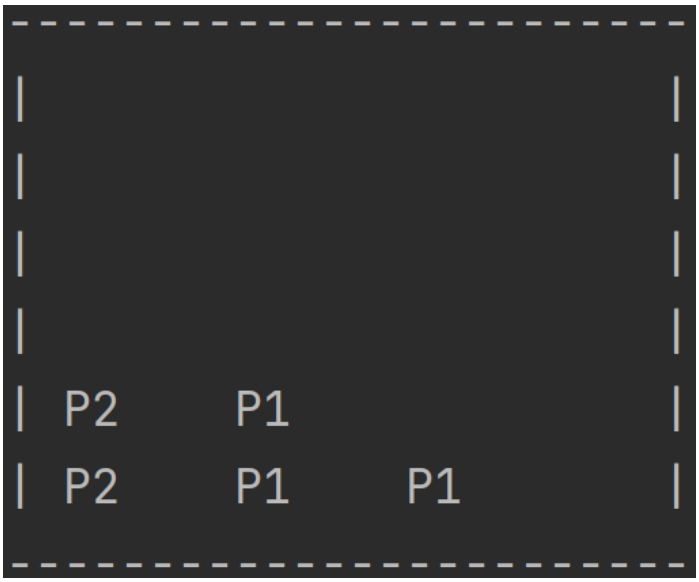
```
def _update_state(self, agent, column):
    """Builds the state of an agent using an action"""
    # use the previous state for invalid move comparison
    old_state = self.states[self.agents[self.agent_name_mapping[agent] -
1]][1].copy()
    # swap 1s and 2s
    mask_1, mask_2 = old_state == 1, old_state == 2
    old_state[mask_1], old_state[mask_2] = 2, 1

    new_state = old_state.copy()
    if not (new_state[0, :, column]).all():  # if not invalid
        last_row = np.where(new_state[0, :, column] > 0)[0]
        try:
            last_row = last_row[0]
        except:
            last_row = self.height
        new_state[0, last_row - 1, column] = 1

    self.states[agent] = np.stack([old_state, new_state])
```

Συμπληρωματικές μέθοδοι της μεθόδου step για το περιβάλλον Connect4.

Ακόμη, υλοποιήθηκε μέθοδος render που αναπαριστά την κατάσταση του περιβάλλοντος στην κονσόλα, παρόμοια με το περιβάλλον Sudoku.

```
def render(self, mode="human", agent: Union[str, None] = None):
    if agent is None:
        agent = self.possible_agents[-1]
    sys.stdout.write('\n' + '-' * (4 * self.width - 4) + '\n')
    for i in range(self.height):
        sys.stdout.write('| ')
        r = self.states[agent][1, 0, i]
        for j in range(self.width):
            cell = r[j]
            if cell == 0:
                sys.stdout.write('   ')
            else:
                if (r[j] == 2 and self.agent_name_mapping[agent]) or \
                        (r[j] == 1 and not
self.agent_name_mapping[agent]):
                    cell_value = 'P1 '
                else:
                    cell_value = 'P2 '
                sys.stdout.write(cell_value)
            sys.stdout.write('')
        sys.stdout.write('|\n')
    sys.stdout.write('-' * (4 * self.width - 4) + '\n')
```

Μέθοδος render για το περιβάλλον Connect4.



Εικόνα 72: Αναπαράσταση του παιχνιδιού μέσω της μεθόδου render. Ο παίκτης 1 συμβολίζεται με P1 ενώ ο παίκτης 2 με P2.

Επιπλέον, μια δεύτερη διανυσματοποιημένη και υπολογιστικά παράλληλη εκδοχή, με 8 εσωτερικά αντίγραφα του περιβάλλοντος, δημιουργήθηκε με σκοπό την επιτάχυνση της εκπαίδευσης των πρακτόρων.

```
def Connect4Env(width: int = 7, height: int = 6, connect: int = 4):
    return aec_to_parallel(_connect4(width, height, connect))


def VecConnect4Env(n_envs: int = 8,
                   width: int = 7,
                   height: int = 6,
                   connect: int = 4):
    parallel_env = Connect4Env(width, height, connect)
    vec_env = pettingzoo_env_to_vec_env_v1(parallel_env)
    return concat_vec_envs_v1(vec_env, n_envs,
base_class='stable_baselines3')
```

Παραλληλοποίηση και διανυσματοποίηση του περιβάλλοντος Connect4.

Για την αναγνώριση της νίκης ενός παίκτη, εφαρμόστηκαν συνελίξεις με 4 πυρήνες, ένα οριζόντιο, έναν κάθετο και 2 διαγώνιους, για τις οποίες το αποτέλεσμα διαιρέθηκε δια του 4, όσα είναι και τα εφαπτόμενα πιόνια τα οποία είναι απαραίτητα για την νίκη. Εάν το αποτέλεσμα για οποιαδήποτε συνέλιξη είναι ίσο με 1, ο παίκτης νικά. Η υλοποίηση αυτή έγινε για υπολογιστικούς λόγους, αφού η Python είναι αρκετά αργή σαν γλώσσα και οι συνελίξεις εφαρμόζονται σε επίπεδο C++, όπως και για λόγους μείωσης κώδικα.

```
def _build_winning_convolutions(self):
    self.convolutions = []
    self.convolutions.append(np.array([[1] * self.connect]))
```

```
        self.convolutions.append(np.array([[1]] * self.connect))
        self.convolutions.append(np.identity(self.connect))
        self.convolutions.append(np.flip(np.identity(self.connect), 0))
```

Δημιουργία συνελίξεων για τον έλεγχο νίκης. Η μέθοδος καλείται κατά την αρχικοποίηση του περιβάλλοντος.

6.3 Υλοποίηση και Εκπαίδευση πρακτόρων

Για την υλοποίηση των νευρωνικών δικτύων που χρησιμοποιήθηκαν για τους πράκτορες χρησιμοποιήθηκε η βιβλιοθήκη PyTorch [63], μια βιβλιοθήκη Βαθιάς Μάθησης, ενώ για την δημιουργία των πολιτικών που εκπαιδεύουν τα δίκτυα αυτά χρησιμοποιήθηκε η έκδοση 1.15.0 της Stable Baselines3, μια βιβλιοθήκη Βαθιάς Ενισχυτικής Μάθησης, βασισμένη στην PyTorch.

Οι πράκτορες εκπαιδεύτηκαν με 2 είδη αλγορίθμων, Proximal Policy Optimization για το Sudoku και Proximal Policy Optimization και Deep Q-learning για το Connect4. Για τους αλγόριθμους αυτούς χρησιμοποιήθηκαν εκδοχές τους που συλλέγουν δεδομένα μεταφέροντας το μοντέλο του αλγορίθμου στην CPU και το εκπαιδεύουν μεταφέροντας το στην GPU[66]. Έτσι, δεν δύναται καθυστέρηση μεταφοράς δεδομένων στην GPU κατά την συλλογή, ενώ ταυτόχρονα αξιοποιείται η δύναμη της GPU κατά την εκπαίδευση.

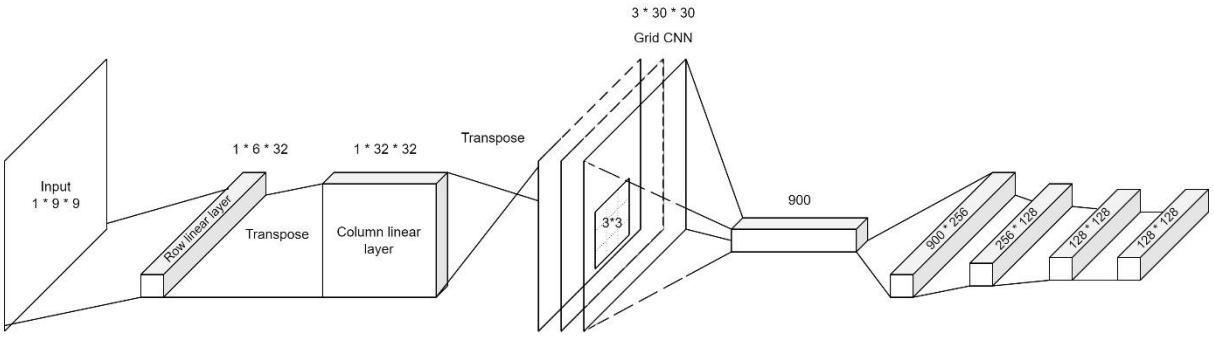
6.3.1 Πράκτορες Sudoku

Για τους πράκτορες Sudoku, 2 αρχιτεκτονικές υλοποιήθηκαν: Ένα μεικτό MLP και συνελικτικό δίκτυο που παρομοιάζει τα MLP mixers, ως ένα βαθμό, και ένα συνελικτικό δίκτυο. Οι πράκτορες εκπαιδεύτηκαν με την χρήση του αλγορίθμου PPO, ενώ οι αρχιτεκτονικές δικτύων Ηθοποιού και Κριτή είναι πανομοιότυπες, χωρίς τη χρήση εξαγωγέα χαρακτηριστικών.

Το “Mixer” δίκτυο βασίστηκε πάνω στην λογική του ίδιου του Sudoku, με 3 σημαντικά στρώματα να χρησιμοποιούνται αρχικά:

- Ένα γραμμικό στρώμα 32 νευρώνων που υπολογίζει την έξοδο για τις σειρές του παιχνιδιού. Έτσι κωδικοποιείται η πληροφορία σχετικά με τους αριθμούς που υπάρχουν στην κάθε γραμμή.
- Ένα γραμμικό στρώμα 32 νευρώνων που υπολογίζει την έξοδο για τις στήλες του παιχνιδιού. Αυτό επιτυγχάνεται με μια αναστροφή στις διαστάσεις των γραμμών και σειρών. Έτσι κωδικοποιείται η πληροφορία για τις στήλες.
- Ένα συνελικτικό στρώμα με 3 πυρήνες 3*3. Πριν το συνελικτικό στρώμα οι γραμμές με τις στήλες αναστρέφονται ώστε να επανέλθουν οι διαστάσεις στο φυσιολογικό. Έτσι κωδικοποιείται η πληροφορία σχετικά με τα υποπλέγματα.

Μετά τα 3 αυτά στρώματα, η έξοδος του συνελικτικού μετατρέπεται σε διάνυσμα και 4 γραμμικά στρώματα ακολουθούν, με 256, 128, 128 και 128 νευρώνες. Ανάμεσα σε όλα τα γραμμικά στρώματα μια συνάρτηση ReLU χρησιμοποιείται, ενώ πριν την είσοδο του συνελικτικού, χρησιμοποιείται μια συνάρτηση Leaky ReLU για την εισαγωγή μη γραμμικότητας.

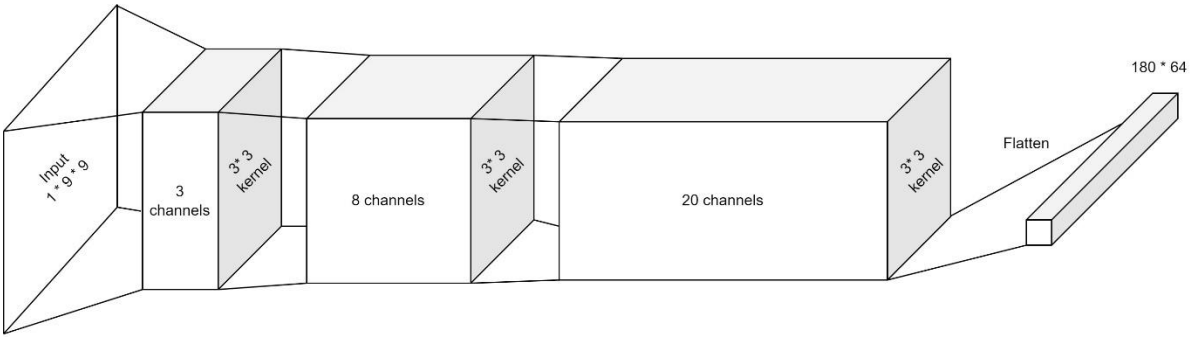


Εικόνα 73: "Mixer" δίκτυο για το Sudoku. Το δίκτυο είναι πανομοιότυπο για τον Ηθοποιό και τον Κριτή.

```
class SudokuMixerNetwork(nn.Module):
    def __init__(self):
        super(SudokuMixerNetwork, self).__init__()

        self.linear = nn.Linear(9, 32)
        self.linear_transpose = nn.Linear(9, 32)
        self.cnn = nn.Conv2d(1, 1, 3, padding='valid')
        self.flat = nn.Flatten()
        self.linear1 = nn.Linear(900, 256)
        self.linear2 = nn.Linear(256, 128)
        self.linear3 = nn.Linear(128, 128)
        self.linear_out = nn.Linear(128, 128)

    def forward(self, x: th.Tensor) -> th.Tensor:
        out = F.relu(self.linear(x)).transpose(2, 3)  # column
        out = F.relu((self.linear_transpose(out))).transpose(2, 3)  # row
        out = F.leaky_relu(self.cnn(out))  # grid
        out = F.relu(self.flat(self.linear1(out)))
        out = F.relu(self.linear2(out))
        out = F.relu(self.linear3(out))
        return F.relu(self.linear_out(out))  # latent


class SudokuMixerACNetwork(nn.Module):

    def __init__(self):
        super(SudokuMixerACNetwork, self).__init__()

        self.actor = SudokuMixerNetwork()
        self.critic = SudokuMixerNetwork()

        self.latent_dim_pi, self.latent_dim_vf = 256, 256

    def forward(self, x: th.Tensor) -> Tuple[th.Tensor, th.Tensor]:
        return self.actor(x), self.critic(x)

    def forward_actor(self, x: th.Tensor) -> th.Tensor:
        return self.actor(x)

    def forward_critic(self, x: th.Tensor) -> th.Tensor:
        return self.critic(x)
```

Κώδικας για την δημιουργία δικτύου Ηθοποιού και Κριτή για το Sudoku με την χρήση του "Mixer" δικτύου.

Το συνελικτικό δίκτυο που χρησιμοποιήθηκε είναι ένα πολύ απλό δίκτυο, με 3 συνελικτικά στρώματα χωρίς max pooling, λόγω της ήδη μικρής εικόνας της παρατήρησης, και ένα γραμμικό δίκτυο στο τέλος. Τα συνελικτικά στρώματα αποτελούνται από 3, 8 και 20 πυρήνες 3*3 αντιστοίχως, ενώ το γραμμικό στρώμα αποτελείται από 64 νευρώνες. Μεταξύ όλων των στρωμάτων χρησιμοποιήθηκε η μη γραμμική συνάρτηση ReLU.



Εικόνα 74: Συνελικτικό δίκτυο για το Sudoku. Το δίκτυο είναι πανομοιότυπο για τον Ηθοποιό και τον Κριτή.

```
class SudokuCnnetwork(nn.Module):
    def __init__(self):
        super(SudokuCnnetwork, self).__init__()

        self.cnn1 = nn.Conv2d(1, 3, 3)
        self.cnn2 = nn.Conv2d(3, 8, 3)
        self.cnn3 = nn.Conv2d(8, 20, 3)
        self.flat = nn.Flatten()
        self.lin_out = nn.Linear(180, 64)

    def forward(self, x: th.Tensor) -> th.Tensor:
        out = F.relu(self.cnn1(x))
        out = F.relu(self.cnn2(out))
        out = F.relu(self.cnn3(out))
        out = F.relu(self.flat(out))
        return F.relu((self.lin_out(out)))


class SudokuCnnACNetwork(nn.Module):
    def __init__(self):
        super(SudokuCnnACNetwork, self).__init__()

        self.actor, self.critic = SudokuCnnetwork(), SudokuCnnetwork()

        self.latent_dim_pi, self.latent_dim_vf = 64, 64

    def forward(self, x: th.Tensor) -> Tuple[th.Tensor, th.Tensor]:
        return self.actor(x), self.critic(x)

    def forward_actor(self, x: th.Tensor) -> th.Tensor:
        return self.actor(x)

    def forward_critic(self, x: th.Tensor) -> th.Tensor:
        return self.critic(x)
```

Κώδικας για την δημιουργία δικτύου Ηθοποιού και Κριτή για το Sudoku με την χρήση του συνελικτικού δικτύου.

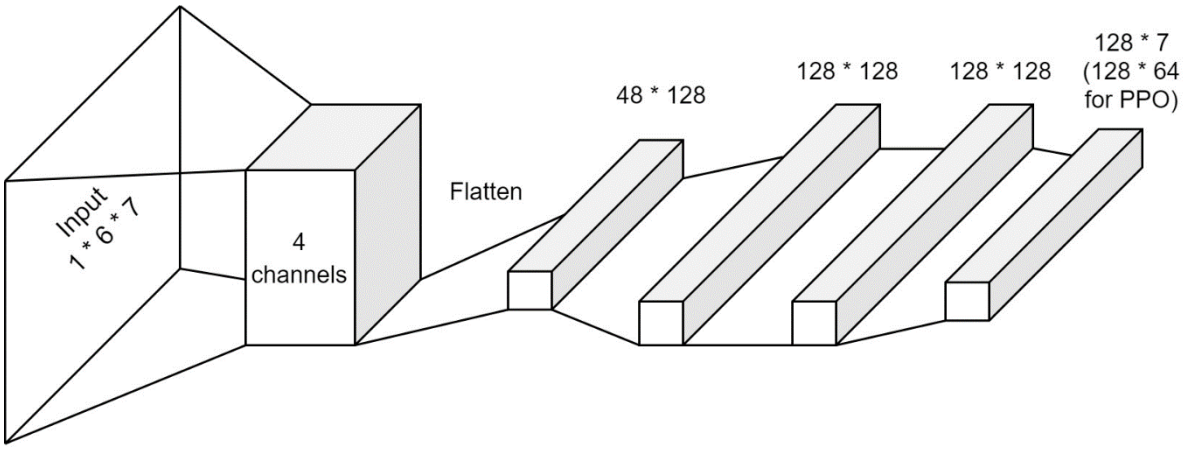
```python
class SudokuMixerACPolicy(ActorCriticPolicy):

    def __init__(self, *args, **kwargs):
        super(SudokuMixerACPolicy, self).__init__(*args, **kwargs)

    def _build_mlp_extractor(self) -> None:
        self.mlp_extractor = SudokuMixerACNetwork()

    # Bypass observation preprocessing and features extractor
    def extract_features(self, obs: th.Tensor) -> th.Tensor:
        return obs.float()  # Handle Double type tensor error


class SudokuCnnACPolicy(ActorCriticPolicy):

    def __init__(self, *args, **kwargs):
        super(SudokuCnnACPolicy, self).__init__(*args, **kwargs)

    def _build_mlp_extractor(self) -> None:
        self.mlp_extractor = SudokuCnnACNetwork()

    # Bypass observation preprocessing and features extractor
    def extract_features(self, obs: th.Tensor) -> th.Tensor:
        return obs.float()  # Handle Double type tensor error
```

Κώδικας πολιτικών PPO για το Sudoku.

6.3.2 Πράκτορες Connect4

Ενώ για το Sudoku 2 είδη δικτύων χρησιμοποιήθηκαν, ένα από τα οποία αρκετά περίπλοκο, για το Connect4, ένα απλό μεικτό δίκτυο με συνελικτικά και γραμμικά στρώματα χρησιμοποιήθηκε. Στην περίπτωση του Connect4, πράκτορες εκπαιδεύτηκαν με PPO, αλλά και DQN, με τον Ηθοποιό και Κριτή και το Q-δίκτυο αλλά και Q-δίκτυο στόχο να είναι εξίσου πανομοιότυπα σε αρχιτεκτονική.

Το δίκτυο για τους πράκτορες του Connect4 αποτελείται από ένα συνελικτικό στρώμα 4 πυρήνων 4*4, ώστε να μαθευτούν όσο το δυνατόν καλύτερα οι 4 συνελίξεις για τα 4 εφαπτόμενα πιόνια που κερδίζουν το παιχνίδι, και 4 γραμμικά στρώματα με 128, 128, 128 και 7, για τον αλγόριθμο DQN, ή 64 για τον αλγόριθμο PPO, αντιστοίχως. Ο λόγος που οι νευρώνες του τελευταίου στρώματος αλλάζουν στο DQN είναι πως σε αυτή την περίπτωση το δίκτυο κάνει μια εκτίμηση για τις 7 διαφορετικές στήλες που ο πράκτορας μπορεί να τοποθετήσει ένα πιόνι, ενώ στην περίπτωση της PPO η έξοδος του δικτύου αποτελεί ένα λανθάνων (latent) διάνυσμα, μέσω του οποίου οι πιθανότητες κάθε πράξης υπολογίζονται για τον Ηθοποιό και οι εκτιμήσεις για κάθε πράξη γίνονται για τον Κριτή. Τέλος, ανάμεσα σε κάθε στρώμα εφαρμόζεται η μη γραμμική συνάρτηση ReLU.



Εικόνα 75: Συνελικτικό και MLP δίκτυο για το Connect4. Το δίκτυο είναι πανομοιότυπο για το Q-δίκτυο και το Q-δίκτυο στόχο στον αλγόριθμο DQN, όπως και για τον Ηθοποιό και τον Κριτή στον αλγόριθμο PPO

```
class Connect4CnnMlpNetwork(nn.Module):
    def __init__(self):
        super(Connect4CnnMlpNetwork, self).__init__()

        self.cnn = nn.Conv2d(1, 4, 4, padding='valid')
        self.flat = nn.Flatten()
        self.linear1 = nn.Linear(48, 128)
        self.linear2 = nn.Linear(128, 128)
        self.linear3 = nn.Linear(128, 128)
        self.linear_out = nn.Linear(128, 64)

    def forward(self, x: th.Tensor) -> th.Tensor:
        out = F.relu(self.cnn(x))
        out = F.relu(self.linear1(self.flat(out)))
        out = F.relu(self.linear2(out))
        out = F.relu(self.linear3(out))
        return F.relu(self.linear_out(out))


class Connect4CnnMlpACNetwork(nn.Module):
    def __init__(self):
        super(Connect4CnnMlpACNetwork, self).__init__()
        self.actor, self.critic = Connect4CnnMlpNetwork(),
Connect4CnnMlpNetwork()
        self.latent_dim_pi, self.latent_dim_vf = 64, 64

    def forward(self, x: th.Tensor) -> Tuple[th.Tensor, th.Tensor]:
        return self.actor(x), self.critic(x)

    def forward_actor(self, x: th.Tensor) -> th.Tensor:
        return self.actor(x)

    def forward_critic(self, x: th.Tensor) -> th.Tensor:
        return self.critic(x)


class Connect4CnnMlpQNetwork(QNetwork):
    def __init__(self, *args, **kwargs):
        super(Connect4CnnMlpQNetwork, self).__init__(*args, **kwargs)
        self.q_net = Connect4CnnMlpNetwork()
        self.q_net.linear_out =
```

```
nn.Linear(self.q_net.linear_out.in_features, self.action_space.n)
        self.features_extractor = nn.Identity()
```
Κώδικας για την δημιουργία του Q-δικτύου και του Q-δικτύου στόχου στον αλγόριθμο DQN, καθώς και των δικτύων Ηθοποιού και Κριτή στον αλγόριθμο PPO, για το Connect4.

```
class Connect4CnnMlpACPolicy(ActorCriticPolicy):

    def _build_mlp_extractor(self) -> None:
        self.mlp_extractor = Connect4CnnMlpACNetwork()

    # Bypass observation preprocessing and features extractor
    def extract_features(self, obs: th.Tensor) -> th.Tensor:
        return obs.float()  # Handle Double type tensor error


class Connect4CnnMlpDQNPolicy(DQNPolicy):

    def make_q_net(self) -> QNetwork:
        # Make sure we always have separate networks for features
extractors etc
        net_args = self._update_features_extractor(self.net_args,
features_extractor=None)
        return Connect4CnnMlpQNetwork(**net_args).to(self.device)
```
Κώδικας πολιτικών PPO και DQN για το Connect4.

6.4 Αποτελέσματα

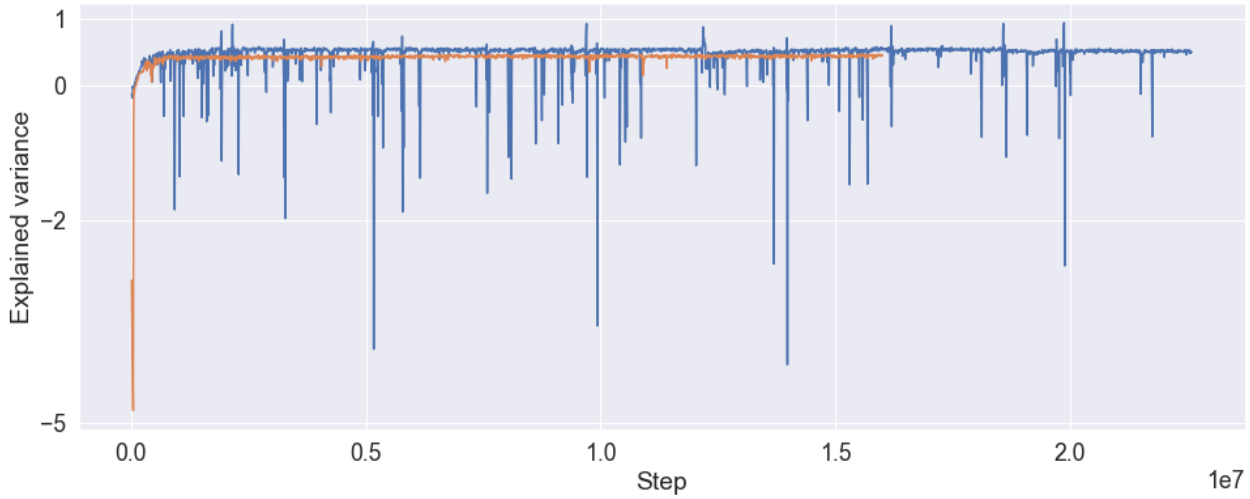
Σε αυτή την ενότητα παρουσιάζονται τα αποτελέσματα των πρακτόρων. Ξεκινούμε με την ανάλυση των αποτελεσμάτων των πρακτόρων για το Sudoku στην υποενότητα 6.4.1 και συνεχίζουμε με την ανάλυση των πρακτόρων για το Connect4 στην 6.4.2.

6.4.1 Sudoku

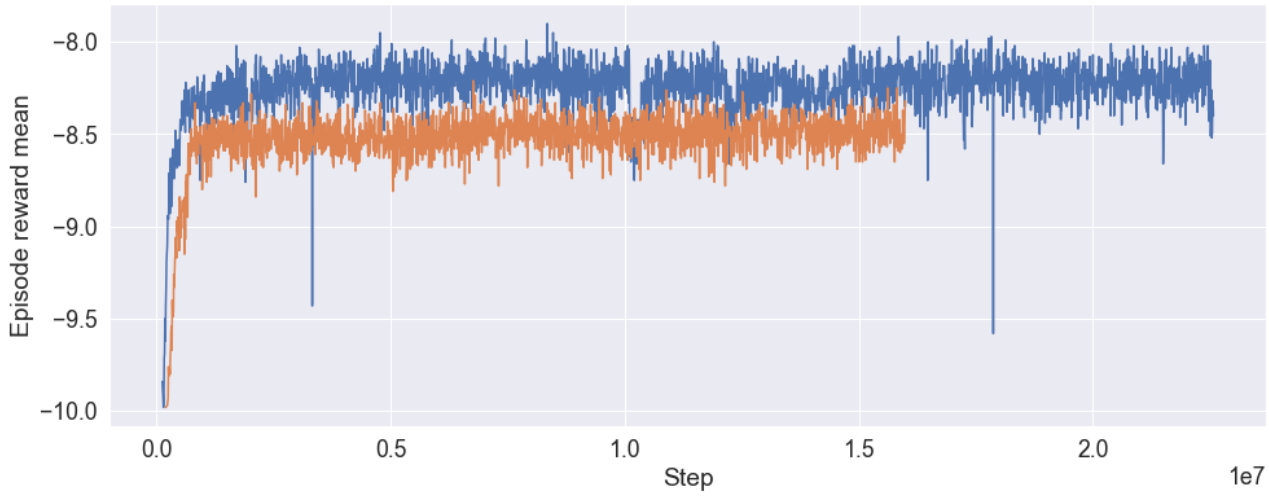
Στην περίπτωση του Sudoku, δύο πράκτορες εκπαιδεύτηκαν: ένας με τη χρήση του δικτύου “Mixer” και ένας με τη χρήση του συνελικτικού δικτύου, για ~15 εκατομμύρια και ~22 εκατομμύρια βήματα αντίστοιχα. Καθένας από τους πράκτορες εκπαιδεύτηκε με:

- Διανυσματοποιημένο περιβάλλον 16 αντιγράφων
- Αλγόριθμο PPO
- Παράμετρο γ ίση με 0.99
- Παραμέτρους policy gradient loss, value function loss και entropy loss ίσες με 1, 0.5 και 0
- Έναν βελτιστοποιητή Adam με βήμα εκπαίδευσης 0.0001
- 512 βήματα κατά την συλλογή δεδομένων για το κάθε αντίγραφο του περιβάλλοντος
- Μέγεθος batch 1024 για την εκπαίδευση
- 10 εποχές εκπαίδευσης για κάθε rollout

Και οι δύο πράκτορες απέτυχαν να λύσουν το περιβάλλον Sudoku, με τον πράκτορα βασισμένο στο συνελικτικό δίκτυο συγκεκριμένα να έχει λιγότερη επιτυχία. Τα αποτελέσματα αυτά είναι εμφανή από τις εικόνες 76 και 77 που αναπαριστούν την μέση επιστροφή των επεισοδίων αλλά και την χαμηλή επεξηγησιμότητα της διασποράς της επιστροφής από τον Κριτή.



Εικόνα 76: Επεξηγήσιμη διασπορά της επιστροφής από τον Κριτή. Η πορτοκαλί γραμμή αναπαριστά τον πράκτορα με το δίκτυο Mixer ενώ η μπλε γραμμή αναπαριστά τον πράκτορα με το συνελικτικό δίκτυο. Έκτοπες τιμές της επεξηγήσιμης διασποράς μικρότερες τις τιμής -5 δεν εμφανίζονται.



Εικόνα 77: Μέση ανταμοιβή των επεισοδίων. Η πορτοκαλί γραμμή αναπαριστά τον πράκτορα με το δίκτυο Mixer ενώ η μπλε γραμμή αναπαριστά τον πράκτορα με το συνελικτικό δίκτυο.

Παρά το γεγονός πως κανένας πράκτορας δεν έδειξε ικανοποιητικά αποτελέσματα, κάτι τέτοιο είναι αναμενόμενο στην περίπτωση του Sudoku, αφού ο αριθμός των παραγόντων που πρέπει να ληφθούν υπόψη είναι αρκετά μεγάλος και το γεγονός ότι κανένα παιχνίδι Sudoku δεν είναι απαραίτητα παρόμοιο με ένα άλλο, με τον αριθμό των δυνατών παιχνιδιών που μπορούν να υπάρξουν στο Sudoku να ανέρχεται σε περίπου ~6.67 εξάκις εκατομμύρια. Ως αποτέλεσμα, το παιχνίδι είναι πολύ δύσκολο να επιλυθεί με κλασσική Ενισχυτική Μάθηση, όπως Q-learning, λόγω του εξαιρετικά μεγάλου αριθμού πιθανών καταστάσεων, αλλά και από Βαθιά Ενισχυτική Μάθηση, όπου προσεγγιστικές λύσεις όπως τα Νευρωνικά Δίκτυα αποτυγχάνουν.[63]

Επιπλέον, απόπειρες έχουν γίνει για την επίλυση του Sudoku με μάθηση με επίβλεψη, Νευρωνικά Δίκτυα και σχετικά ικανοποιητικά αποτελέσματα [64], οι λύσεις όμως των οποίων δεν είναι ποτέ

ακριβείς και απαιτούν την δημιουργία ενός εξαιρετικά βαθέως νευρωνικού δικτύου και την ύπαρξη ενός μεγάλου dataset, ικανού να καλύψει μια πληθώρα περιπτώσεων.

Από την άλλη, αλγόριθμοι όπως γενετικοί αλγόριθμοι και ικανοποίησης περιορισμών, αλλά και άλλες υπολογιστικές λύσεις, όπως ο αλγόριθμος backtracking που περαιτέρω υλοποιήθηκε και μπορεί να χρησιμοποιηθεί για την επίλυση τέτοιων παιχνιδιών, μπορούν να δώσουν απαντήσεις πολύ πιο αποδοτικά αλλά και αποτελεσματικά.

6.4.2 Connect4

Στην περίπτωση του Connect4, 2 πράκτορες εκπαιδεύτηκαν με την χρήση του αλγορίθμου PPO και 3 πράκτορες με την χρήση του αλγορίθμου DQN, χρησιμοποιώντας την ίδια αρχιτεκτονική του μεικτού δικτύου για τον Ηθοποιό, τον Κριτή, το Q-δίκτυο και το Q-δίκτυο στόχο.

Οι πράκτορες PPO εκπαιδεύτηκαν με:

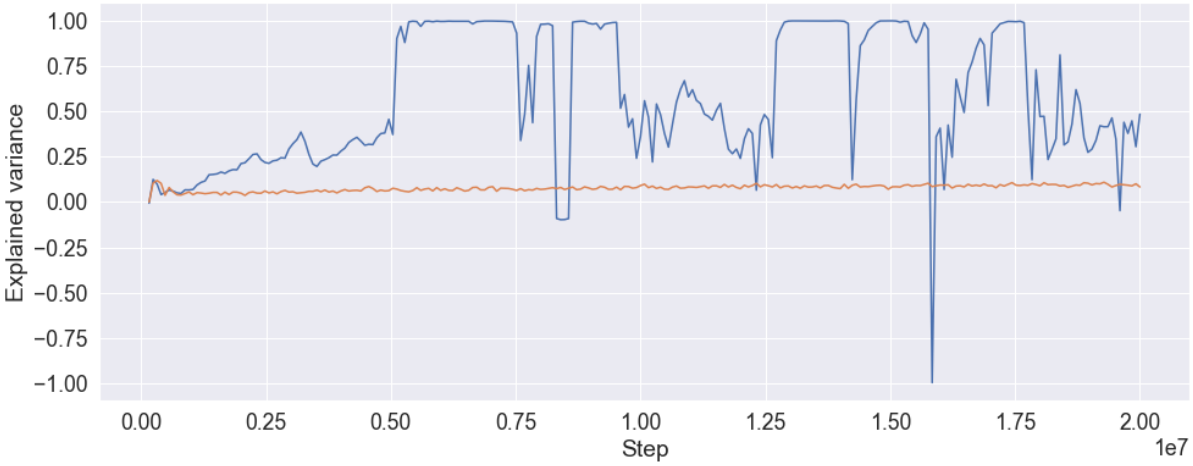
- Διανυσματοποιημένο και υπολογιστικά παράλληλο περιβάλλον 16 αντιγράφων
- Παράμετρο γ ίση με 0.95
- Παραμέτρους policy gradient loss και value function loss ίσες με 1 και 0.5
- Έναν βελτιστοποιητή Adam με βήμα εκπαίδευσης 0.0001
- 5000 βήματα κατά την συλλογή δεδομένων για κάθε αντίγραφο του περιβάλλοντος
- Μέγεθος batch 5000 για την εκπαίδευση
- 10 εποχές εκπαίδευσης για κάθε rollout
- 20 εκατομμύρια βήματα σε σύνολο

Για τον πρώτο πράκτορα, η παράμετρος entropy loss ήταν ίση 0, ενώ για τον δεύτερο πράκτορα η παράμετρος ήταν ίση με 0.5, προάγοντας την εξερεύνηση περισσότερων πράξεων και κατ' επέκταση καταστάσεων του περιβάλλοντος.

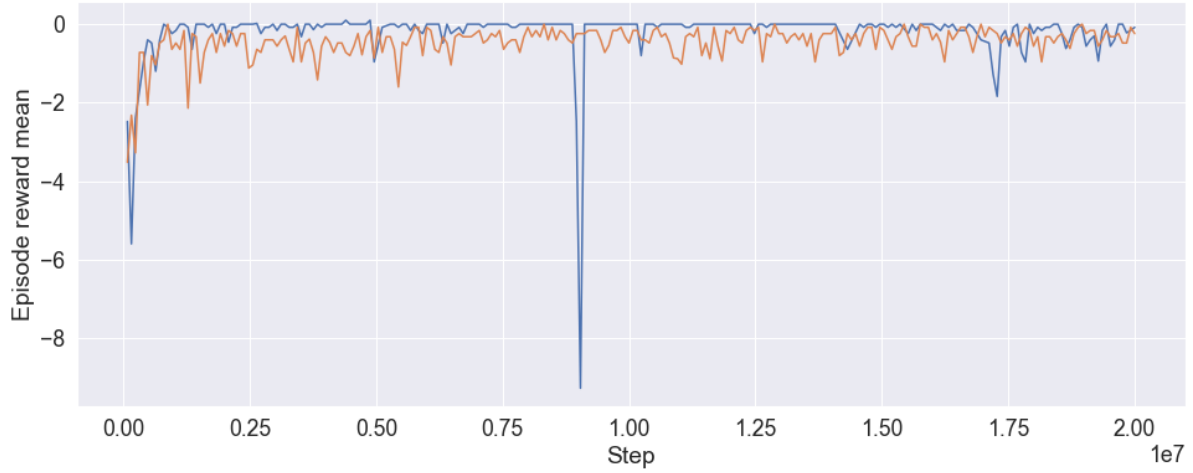
Οι πράκτορες DQN εκπαιδεύτηκαν με:

- Διανυσματοποιημένο και υπολογιστικά παράλληλο περιβάλλον 16 αντιγράφων
- Παράμετρο γ ίση με 0.95
- Έναν βελτιστοποιητή Adam με βήμα εκπαίδευσης 0.0001
- Μέγεθος batch 5000 για την εκπαίδευση
- Μέγεθος replay buffer 1 εκατομμύριου samples
- 5000 βήματα κατά την συλλογή δεδομένων για κάθε αντίγραφο του περιβάλλοντος
- 1 εποχή εκπαίδευσης
- Παράμετρο εξερεύνησης που ξεκινά από 1 και φθίνει γραμμικά σε 0.05 έως το πέρας του 85% της συνολικής εκπαίδευσης
- Παράμετρο τ ίση με 1 για hard updates
- 1, 1.5 και 2 εκατομμύρια βήματα σε σύνολο

Σε σχέση με το Sudoku, όλοι οι πράκτορες κατάφεραν να μάθουν. Από αυτούς, εκείνοι που εκπαιδεύτηκαν με τον αλγόριθμο PPO έδειξαν καλύτερα αποτελέσματα, με τον πράκτορα με την μεγαλύτερη παράμετρο entropy loss να παρουσιάζει πιο σταθερές εκτιμήσεις και σταθερά ανοδική πορεία, ενώ ο πράκτορας με παράμετρο entropy loss 0 παρουσίασε καλύτερες μετρικές αλλά μεγαλύτερη αστάθεια. Τα αποτελέσματα από την εκπαίδευση των πρακτόρων που εκπαιδεύτηκαν με τον αλγόριθμο PPO παρουσιάζονται στις εικόνες 78 και 79.

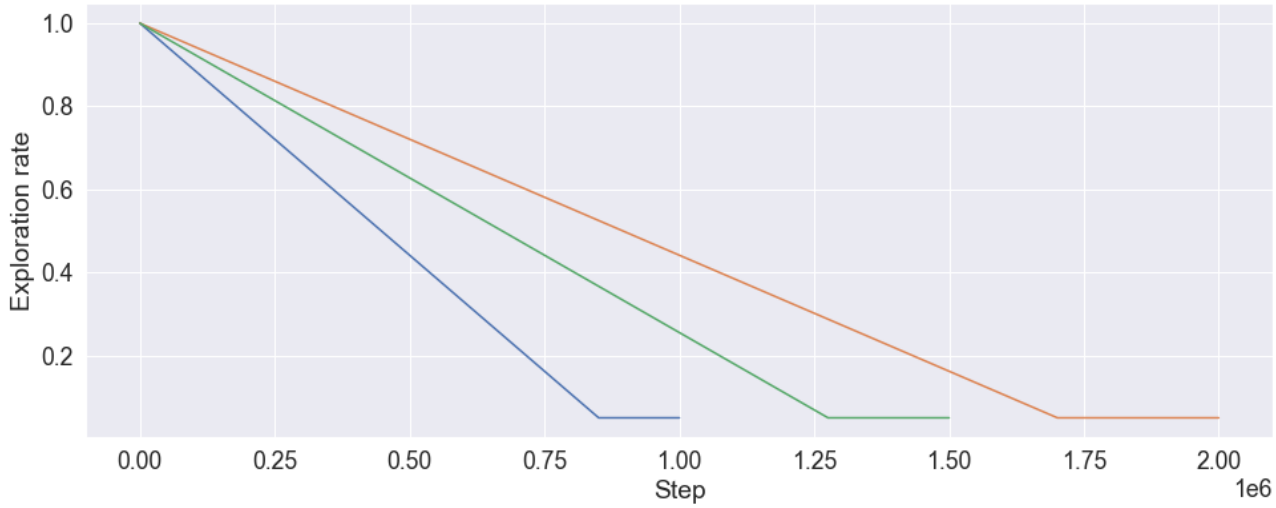


Εικόνα 78: Επεξηγήσιμη διασπορά της επιστροφής από τον Κριτή για τους πράκτορες PPO. Η μπλε γραμμή αναπαριστά τον πράκτορα με παράμετρο entropy loss 0 ενώ η πορτοκαλί γραμμή αναπαριστά τον πράκτορα με παράμετρο entropy loss 0.5.

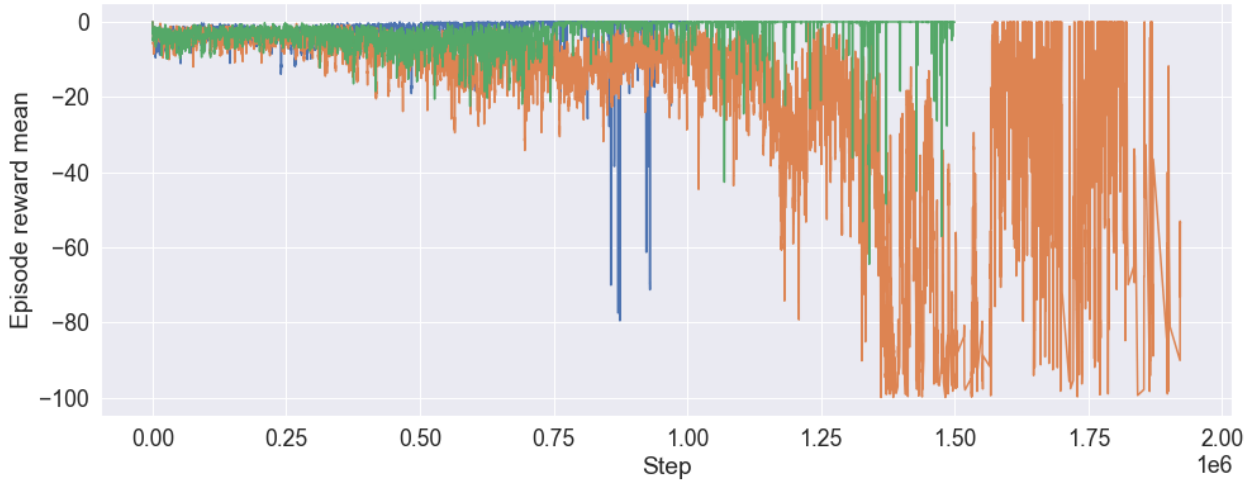


Εικόνα 79: Μέση ανταμοιβή των επεισοδίων για τους πράκτορες PPO. Η μπλε γραμμή αναπαριστά τον πράκτορα με παράμετρο entropy loss 0 ενώ η πορτοκαλί γραμμή αναπαριστά τον πράκτορα με παράμετρο entropy loss 0.5. Το βέλτιστο είναι το μηδέν. Έκτοπες τιμές της μέσης ανταμοιβής των επεισοδίων μικρότερες της τιμής -10 δεν εμφανίζονται.

Από τους πράκτορες DQN, ο πράκτορας των 1.5 εκατομμυρίων βημάτων ξεχώρισε, συγκλίνοντας σε μία καλύτερη πολιτική, αξιοποιώντας ταυτοχρόνως γνώση από πολλές καταστάσεις. Ως σύγκριση, ο πράκτορας των 2 εκατομμυρίων βημάτων απέκλινε και παρουσίασε επιδείνωση της πολιτικής του έπειτα από τα 1.2 εκατομμύρια βήματα, το οποίο εξηγείται από την καθυστέρηση της πτώσης της παραμέτρου εξερεύνησης. Τα αποτελέσματα για τους πράκτορες που εκπαιδεύτηκαν με τον αλγόριθμο DQN παρουσιάζονται στις εικόνες 80 και 81.



Εικόνα 80: Παράμετρος εξερεύνησης για τους πράκτορες DQN. Η μπλε γραμμή αναπαριστά τον πράκτορα με 1 εκατομμύρια βήματα, η πράσινη γραμμή τον πράκτορα με 1.5 εκατομμύρια βήματα και η πορτοκαλί γραμμή τον πράκτορα με 2 εκατομμύρια βήματα.



Εικόνα 81: Μέση ανταμοιβή των επεισοδίων για τους πράκτορες DQN. Η μπλε γραμμή αναπαριστά τον πράκτορα με 1 εκατομμύρια βήματα, η πράσινη γραμμή τον πράκτορα με 1.5 εκατομμύρια βήματα και η πορτοκαλί γραμμή τον πράκτορα με 2 εκατομμύρια βήματα. Το βέλτιστο είναι το μηδέν. Έκτοπες τιμές της μέσης ανταμοιβής των επεισοδίων μικρότερες του -100 δεν εμφανίζονται.

Όλοι οι πράκτορες κατόρθωσαν να κατανοήσουν την οριζόντια και κάθετη στοίχιση σε ικανοποιητικό βαθμό. Από την άλλη, οι πράκτορες απέτυχαν να μάθουν διαγώνιες στοιχίσεις, μία δεξιότητα που εξαρτάται κυρίως από τις κινήσεις του αντιπάλου και την ικανότητα του παίκτη.

Παρά το γεγονός ότι όλοι οι πράκτορες έμαθαν, η ικανότητα τους στο παιχνίδι δεν μπορεί διόλου να συγκριθεί με αυτή ενός ανθρώπου. Αυτό μπορεί να εξηγηθεί από τα εξής:

- Οι πράκτορες μετρούν την ποιότητα μιας κίνησης ως προς μια κατάσταση, χωρίς όμως να λαμβάνουν υπόψη παρελθοντικές κινήσεις. Μια τακτική που μπορεί να ενισχύσει την επίδοση των πρακτόρων είναι η χρήση ενός αναδρομικού στρώματος στον πράκτορα. Αυτό προϋποθέτει όμως την τροποποίηση των rollout και replay buffers που συλλέγουν τα δεδομένα του περιβάλλοντος, κάτι το οποίο αποτελεί εσωτερικό κομμάτι της ίδιας της βιβλιοθήκης Stable Baselines3.

- Ο κάθε πράκτορας χειρίζεται και τους 2 παίκτες ταυτόχρονα, με τον κάθε παίκτη να μπορεί να εκτιμήσει την έκβαση της κίνησης που θα κάνει χωρίς να γνωρίζει την κίνηση του άλλου εκ των προτέρων. Το πρόβλημα αυτό ανάγεται στην φύση του περιβάλλοντος, το οποίο είναι μερικώς παρατηρήσιμο, λόγω περιορισμών της βιβλιοθήκης PettingZoo.

Τέλος, για την ανθρώπινη εκτίμηση των πρακτόρων που δημιουργήθηκαν, ένα επιπλέον script δημιουργήθηκε σε Python, μέσω του οποίου ένας άνθρωπος μπορεί να παίξει εναντίον τους. Το script μπορεί να κληθεί με τις ακόλουθες παραμέτρους:

- h: Παρέχει πληροφορίες βοήθειας σχετικά με τον τρόπο χρήσης του script.
- algorithm [PPO, DQN]: Τύπος αλγόριθμου πράκτορα. Μπορεί να είναι PPO ή DQN. Ο αλγόριθμος DQN χρησιμοποιείται ως default.
- model-file: Τοποθεσία μοντέλου πράκτορα. Μπορεί να είναι σχετική ως προς την τοποθεσία από την οποία καλείται το script ή απόλυτη.
- n-games: Αριθμός παιχνιδιών με τον πράκτορα. Η default τιμή είναι 1.
- first [human, agent]: Εάν θα ξεκινήσει ο πράκτορας ή ο άνθρωπος πρώτος. Η default τιμή είναι agent.

Καθ' όλη την διάρκεια του παιχνιδιού εναντίων του πράκτορα, η κατάσταση του περιβάλλοντος αναπαρίσταται στην κονσόλα.

```
import argparse
from functools import partial
from time import sleep

from stable_baselines3 import DQN, PPO

from envs.connect4 import _connect4

usage = "\narguments:\n" + \
        "-h: help\n" + \
        "--algo [PPO, DQN]: algorithm to be used, defaults to DQN\n" + \
        "--model-file [<some_model_file.zip>]: model file of the agent\n" +
\
        "--n-games [<integer number of games>]: defaults to 1\n" + \
        "--first [human, agent]: human or agent first, defaults to agent"

parser = argparse.ArgumentParser(usage=usage,
                                 description="Play Connect4 against a
trained agent.",
                                 epilog="Example of usage:\n"
                                        "python play_connect4.py --algo PPO
--model-file model_1000000_steps.zip "
                                        "--n-games 1 --first agent")

parser.add_argument("--algo", default="DQN", choices=["PPO", "DQN"])
parser.add_argument("--model-file", required=True)
parser.add_argument("--n-games", default=1, type=int)
parser.add_argument("--first", default="agent", choices=["agent", "human"])

if __name__ == '__main__':
    def play_agent(player: str):
        done = False

        sleep(2)
```

```
        agent_action = model.predict(aec_env.states[player][1],
deterministic=True)[0].item()
        print("\nAgent plays: " + str(agent_action + 1))
        aec_env._update_state(player, agent_action)
        aec_env.render(agent=player)

        if aec_env._check_win(aec_env.states[player][1, 0]):
            print("Agent wins!")
            done = True
        elif (aec_env.states["player_0"][1] > 0).all():
            print("Game ends in draw.")
            done = True

        return done


    def play_human(player: str):
        done = False

        while True:
            human_action = input("\nHuman plays: ")
            try:
                human_action = int(human_action) - 1
            except:
                print("Action should be an integer between 1 and " +
str(aec_env.width))
                continue
            if human_action > (aec_env.width - 1) or human_action < 0:
                print("Action should be between 1 and " +
str(aec_env.width))
                continue
            break
        aec_env._update_state(player, human_action)
        aec_env.render(agent=player)

        if aec_env._check_win(aec_env.states[player][1, 0]):
            print("Human wins!")
            done = True
        elif (aec_env.states[player][1] > 0).all():
            print("Game ends in draw.")
            done = True

        return done


    args = parser.parse_args()

    n_games = args.n_games
    algorithm_cls = eval(args.algo)
    model_file = args.model_file
    first = args.first

    if first == "agent":
        plays = [partial(play_agent, "player_0"), partial(play_human,
"player_1")]
        start_msg = "Agent goes first as P1, human goes second as P2"
    else:
        plays = [partial(play_human, "player_0"), partial(play_agent,
"player_1")]
        start_msg = "Human goes first as P1, agent goes second as P2"
```

```
    aec_env = _connect4()

    custom_objects = {'n_envs': 1}
    model = algorithm_cls.load(model_file, device="cpu",
custom_objects=custom_objects)

    print(start_msg)

    for n in range(n_games):
        aec_env.reset()
        print("Game " + str(n + 1) + "/" + str(n_games) + " begins!")
        sleep(2)
        aec_env.render()

        while True:
            if plays[0]():
                break
            if plays[1]():
                break
```

Κώδικας script μέσω του οποίου ένας άνθρωπος μπορεί να παίξει εναντίον ενός πράκτορα.

Κεφάλαιο 7ο: Συμπεράσματα και μελλοντικές βελτιώσεις

Ο βασικός στόχος της διπλωματικής εργασίας ήταν η υλοποίηση της εκπαίδευση πρακτόρων ενισχυτικής μάθησης σε περιβάλλον παιχνιδιού με όσο το δυνατόν μεγαλύτερη απόδοση της εκπαίδευσης του πράκτορα και της απόδοσής του. Στα πλαίσια της εργασίας, όπως προαναφέρθηκε, υλοποιήθηκε και εκπαιδεύτηκε ο πράκτορας με τη χρήση των PPO και DQN αλγορίθμων της βαθιάς ενισχυτικής μάθησης. Κατά τη διάρκεια της εκπόνησης της εργασίας και ειδικότερα της εκπαίδευσης των αλγορίθμων, εξετάστηκαν διάφορες παραλλαγές τόσο του περιβάλλοντος όσο και των πρακτόρων σε μία προσπάθεια βελτιστοποίησης των υφιστάμενων αλγορίθμων. Διαπιστώθηκε η αστάθεια που χαρακτηρίζει τους αλγορίθμους αυτούς, αλλά και τα μεγάλα περιθώρια βελτίωσης τους. Με μικρές αλλαγές στη δομή των πρακτόρων, που τονίζονται στις παρατηρήσεις της υλοποίησης, επιταχυνόταν σημαντικά η βελτίωση της συμπεριφοράς του πράκτορα. Παρόλα αυτά στο Sudoku, σημειώθηκε μια δύσκολη σχετικά εκπαίδευση του πράκτορα καθώς και χαμηλές επιδόσεις στο παιχνίδι, παρόλο που υπήρξε προσπάθεια αλλαγής κώδικα ή και αλγορίθμου. Παρά την προσπάθεια και τις δοκιμές με διάφορες αρχιτεκτονικές, οι περισσότεροι μας πράκτορες εν τέλει απέτυχαν να μάθουν με αξιόπιστο τρόπο να νικούν τα παιχνίδια, κάτι που εξηγείται λεπτομερώς από την φύση των παιχνιδιών, τις ιδιότητες των περιβαλλόντων, αλλά και τις αδυναμίες της βιβλιοθήκης που χρησιμοποιήθηκε. Εν τέλει, παρόλο που οι πράκτορες δεν ήταν ιδιαίτερα αποδοτικοί και πολλές υπολογιστικές μέθοδοι δύνανται για την επίλυση των παιχνιδιών αυτών, το επίκεντρο αυτής της διπλωματικής ήταν η εντριβή με την Βαθιά Ενισχυτική Μάθηση και τους αλγορίθμους της, όπως και την μεθοδολογία υλοποίησης περιβαλλόντων για αυτή αλλά και αντιστοίχων πρακτόρων. Όμως, υιοθετώντας καλύτερη πολιτική, και στα δύο παιχνίδια, κατάφερα να υπάρξουν καλύτερες επιδόσεις καθώς και περισσότερες πιθανότητες επίτευξης κάποιας μεγαλύτερης συνολικής ανταμοιβής.

Στο σημείο αυτό, παρουσιάζεται μία σειρά από προτάσεις που αποσκοπούν στην ανάπτυξη ενός αποδοτικότερου για περιορισμένο χρονικό διάστημα εκπαίδευσης πράκτορα. Σύμφωνα με τον αλγόριθμο Rainbow, το αρχικό στάδιο ευνοεί τον συνδυασμό των εκδόσεων του αλγορίθμου DQN. Αξιοποιούνται με αυτόν τον τρόπο τα πλεονεκτήματα καθεμίας παραλλαγής προς το κοινό όφελος. Επίσης, συνιστάται η παραλληλοποίηση των αλγορίθμων όσο είναι εφικτό. Πιο συγκεκριμένα, ο ίδιος παράγοντας εφαρμόζεται ταυτόχρονα σε πολλά περιβάλλοντα, με αποτέλεσμα να λαμβάνονται πολλές παρατηρήσεις ταυτόχρονα σε κάθε φάση. Προκειμένου να επιλυθεί το δίλημμα εξερεύνηση ή εκμετάλλευση, είναι επίσης κατανοητό για κάθε οικοσύστημα να τηρεί μια ξεχωριστή πολιτική εξερεύνησης. Ακόμη καλό θα ήταν για τη βελτίωση της εργασίας μου, να χρησιμοποιούταν καλύτερο hardware, γιατί εκεί ίσως άξιζε να μελετηθούν πιο σύνθετες υλοποιήσεις πρακτόρων.

Συμπερασματικά, είναι εφικτό να γίνει διάκριση μεταξύ μιας αλληλεπίδρασης και της εκπαίδευσης ενός πράκτορα. Αυτές οι δύο διαδικασίες μπορούν να εκτελεστούν ταυτόχρονα, αξιοποιώντας στο μέγιστο τους διαθέσιμους πόρους επεξεργασίας. Ο απώτερος στόχος της δημιουργίας ενός πράκτορα που θα αποκτήσει τη μέγιστη δυνατή συνολική ανταμοιβή σε μια περιορισμένη περίοδο εκπαίδευσης παραμένει ο ίδιος.

Κεφάλαιο 8ο: Βιβλιογραφία

[1] Kevin R. McKee, Joel Z. Leibo, Charlie Beattie , Richard Everett,"Quantifying the effects of environment and population diversity in multi-agent reinforcement learning", *Autonomous Agents and Multi-Agent Systems*,30 Jul 2021

[2] PapaxristouNikolaos," Reinforcement Learning in Two Person Games Application to Backgammon ",p. 1 ,*Master's Thesis of the Department of Applied Informatics Specialization in Computer Systems*, June 2010

[3] Ioannis Rexakis, " Directed Exploration of Policy Space in Reinforcement Learning*", School of Electrical and Computer Engineering Technical University of Crete, Greece,* 2018

[4] Marios P. Blaxogiannopoulou, "*Deep reinforcement learning using methods reward token-based exploration curiosity",*Andreas Georgios Stafulopatis School of Electrical and Computer EngineeringIT and Computer Technology Sector, October 2021

[5] Simon S. Haykin,"*Neural networks and learning machines",* Third. Pearson Education, 2009

[6] Kate Strachnyi,"*Brief History of Neural Networks"*, online: https://medium.com/analytics-vidhya/brief-history-of-neural-networks-44c2bf72eec , accessed: 23 July 22

[7] Jakob N. Foerster, "*Deep Multi-Agent Reinforcement Learning",* Magdalen College University of Oxford, 2018

[8] John Ruggles. "*Locomotive steam-engine for rail and other roads*",US Patent 1. July 1836

[9] Donald Waterman. *"A guide to expert system*s". In: (1986)

[10] Marti A. Hearst et al. "*Support vector machines*". In: IEEE Intelligent Systems and their applications 13.4 (1998), pp. 18–28.

[11]IBM Cloud Education, "*Neural Networks",* online: https://www.ibm.com/cloud/learn/neural-networks , accessed: 25 July 22

[12] Victor Zhou, "*Machine Learning for Beginners: An Introduction to Neural Networks*", Towards Data Science, Mar 5, 2019

[13] Tushar Gupta,"*Deep Learning: Feedforward Neural Network",* online: https://towardsdatascience.com/deep-learning-feedforward-neural-network-26a6705dbdc7 , accessed: 30 June 2022

[14] Simplilearn,*" What is Perceptron: A Beginners Guide for Perceptron"*,online: https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron?fbclid=IwAR0FVnTHPLBQ3ckPkguH9QUi7_PZ_6GBML1mDff8fICuDBtpuLk4gRfrpqU, accessed: 31 July2022

[15] Pavan Vadapalli, *"Biological Neural Network: Importance, Components & Comparison"*, online:https://www.upgrad.com/blog/biological-neural-network/, accessed: 1 August 2022

[16] Kate Skitikova,*"TWO EFFECTIVE METHODS ON PROTECTING ONE'S HOUSE & THEIR COMPARISON"*, V.N. Karazin Kharkiv National University Journal., 14 Mat 2020

[17] Jahnavi Mahanta, *"Introduction to Neural Networks, Advantages and Applications"*, Towards Data Science, 10 July 2017

[18] Navdeep Singh Gill, "Artificial Neural Networks Applications and Algorithms", online: https://www.xenonstack.com/blog/artificial-neural-network-applications, accessed: 6 August 2022

[19]wikipedia,*"Neuron*",online:https://el.wikipedia.org/wiki/%CE%9D%CE%B5%CF%85%CF%81%CF%8E%CE%BD%CE%B1%CF%82, accessed : 6 August 2022

[20] Prashant Sharma, *"Feedforward Neural Network: Its Layers, Functions, and Importance*", Data Science Blogathon, 28 January 2022

[21] Ramon Quiza,*"Sample of a feed-forward neural network*", online: https://www.researchgate.net/figure/Sample-of-a-feed-forward-neural-network_fig1_234055177, accessed: 7 August 2022

[22] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola, *"Dive into Deep Learning*", 30 July 2022

[23] Chris Nicholson, *"A Beginner's Guide to Multilayer Perceptrons (MLP)*",online:https://wiki.pathmind.com/multilayer-perceptron", accessed: 7 August 2022

[24] Sumit Saha, *"A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*", Towards Data Science, 15 December 2018

[25] Christopher Thomas BSc Hons. MIAP, *"An introduction to Convolutional Neural Networks*",Towards Data Science,27 May 2019

[26] Utkarsh Ankit, *"Transformer Neural Network: Step-By-Step Breakdown of the Beast*",Towards Data Science, 25 April 2020

[27] Mor Geva, Roei Schuster, Jonathan Berant, Omer Levy, *"Transformer Feed-Forward Layers Are Key-Value Memories*",*Blavatnik School of Computer Science, Tel-Aviv University,Allen Institute for Artificial Intelligence,Cornell Tech,* 5 September 2021

[28] Birtsionis Gkalinikis Nikolaos, *"Σχεδίαση βαθιών νευρωνικών δικτύων για το πρόβλημα του επιμερισμού ηλεκτρικής ισχύος*",Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, 2020

[29] Pavan Vadapalli, *"Introduction to Recursive Neural Network: Concept, Principle & Implementation*", online: https://www.upgrad.com/blog/introduction-to-recursive-neural-network/, accessed:13 August 2022

[30] Rohit Sharma, *"Recurrent Neural Network in Python: Ultimate Guide for Beginners*", online: https://www.upgrad.com/blog/recurrent-neural-network-in-python/, accessed:13 August 2022

[31] Simeon Kostadinov, *"Understanding GRU Networks",*Towards Data Science,16 December 2017

[32] Saul Dobilas, *"GRU Recurrent Neural Networks — A Smart Way to Predict Sequences in Python",*Towards Data Science,22 February 2022

[33] Vijaysinh Lendave, *"LSTM Vs GRU in Recurrent Neural Network: A Comparative Study",* Developers Corner, 28 August 2021

[34] IoannisPartalas , *"Μέθοδοι Ενισχυτικής Μάθησης σε Συστήματα Πρακτόρων",*Aristotleuniversity of Thessaloniki, 29 September 2009

[35] Joseph M. Carew, *"reinforcement learning",*online https://www.techtarget.com/searchenterpriseai/definition/reinforcement-learning, accessed: 14 August 2022

[36]Prateek Bajaj, *"Reinforcement Learning",*online:https://www.geeksforgeeks.org/what-is-reinforcement-learning, accessed: 15 August 2022

[37] Ankit Choudhary, *"Nuts & Bolts of Reinforcement Learning: Model Based Planning using Dynamic Programming",*online: https://www.analyticsvidhya.com/blog/2018/09/reinforcement-learning-model-based-planning-dynamic-programming/ , accessed: 15 August 2022

[38] Rohan Jagtap, *"Dynamic Programming in RL",*Towards Data Science, 30 December 2020

[39]Nishant Rana, *"Overlapping substructure vs overlapping sub problems",* online :https://www.codingninjas.com/codestudio/library/overlapping-substructure-vs-overlapping-sub-problems, accessed: 15 August 2022

[40]Raghuveer Bhandarkar, *"Policy Iteration in RL: A step by step illustration",* Towards Data Science, 25 March 2020

[41]A. Aylin Tokuç, *"Value Iteration vs. Policy Iteration in Reinforcement Learning",* online https://www.baeldung.com/cs/ml-value-iteration-vs-policy-iteration , accessed: 15 August 2022

[42] Rohan Jagtap, *"Understanding Markov Decision Process (MDP)",* Towards Data Science, 27 September 2020

[43] Abhishek Sharma 44, *"Markov Decision Process",* online: https://www.geeksforgeeks.org/markov-decision-process/ , accessed: 17 August 2022

[44] Dimitri P. Bertsekas *"Dynamic Programming and Optimal Control 3rd Edition, Volume II",* Massachusetts Institute of Technology, 11 November 2011

[45] Jordi TORRES.AI, *"The Value Iteration Algorithm",* Towards Data Science, 14 June 2020

[46] Baijayanta Roy, *"Monte Carlo Learning",* Towards Data Science, 12 September 2019

[47] Sagi Shaier, *"Monte Carlo Methods",* Towards Data Science, 20 November 2020

[48] Ram Sagar, *"On-Policy VS Off-Policy Reinforcement Learning",* Developers Corner, 11 April 2020

[49] Viet Hoang Tran Duong, *"Intro to reinforcement learning: temporal difference learning", SARSA vs. Q-learning",* Towards Data Science, 24 February 2021

[50] Vaibhav Kumar, "Reinforcement learning: Temporal-Difference, SARSA, Q-Learning & Expected SARSA in python", Towards Data Science, 20 March 2019

[51] Sagi Shaier, "Temporal-Difference (TD) Learning", Towards Data Science, 20 November 2020

[52] Baijayanta Roy, "Temporal-Difference (TD) Learning", Towards Data Science, 12 September 2019

[53] Chris Nicholson, "A Beginner's Guide to Deep Reinforcement Learning",online: https://wiki.pathmind.com/deep-reinforcement-learning, accessed: 25 August 2022

[54] Jordi TORRES.AI, "A gentle introduction to Deep Reinforcement Learning", Towards Data Science, 15 May 2020

[55] Shaked Zychlinski, "Qrash Course II: From Q-Learning to Gradient Policy & Actor-Critic in 12 Minutes", Towards Data Science, 24 November 2019

[56] Chris Yoon, “Understanding Actor Critic Methods and A2C”, Towards Data Science, 6 February 2019

[57] Mike Wang, “Deep Q-Learning Tutorial: minDQN”, Towards Data Science, 18 November 2020

[58] Ziad SALLOUM, “Double Q-Learning, the Easy Way”, Towards Data Science, 6 December 2018

[59] Dhanoop Karunakaran, “The Actor-Critic Reinforcement Learning algorithm”, Towards Data Science, 30 September 2020

[60] Chintan Trivedi, “Proximal Policy Optimization Tutorial (Part 1/2: : Actor-Critic Method)) ”, Towards Data Science, 12 August 2019

[61] Chintan Trivedi, “Proximal Policy Optimization Tutorial (Part 2/2: GAE and PPO loss) ”, Towards Data Science, 12 August 2019

[62] Takuma Seno, “Welcome to Deep Reinforcement Learning Part 1 : DQN”, Towards Data Science, 20 October 2017

[63]Anav Mehta, “Reinforcement Learning For Constraint Satisfaction Game Agents”, Cupertino High School, Cupertino, CA, USA,pp 11-12

[64]Charles Akin-David, Richard Mantey, “Solving Sudoku with Neural Networks”, online: https://cs230.stanford.edu/files_winter_2018/projects/6939771.pdf, accessed: 28 August 2022

[65] online: https://github.com/openai/gym, accessed: 20 April 2022

[66] online: https://www.github.com/Enkhai/sb3-device-alternating, accessed: 25 April 2022

[67] online: https://github.com/DLR-RM/stable-baselines3, accessed:1 May 2022