ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Η/Υ

ΠΟΛΥΤΕΧΝΕΙΟ
ΚΡΗΤΗΣ /
TECHNICAL
UNIVERSITY
OF CRETE

ΠΛΗ 423: Reinforcement Learning

Εξαμινιαία Εργασία:
Εφαρμογές Ενισχυτικής Μάθησης σε Απλοποιημένο παιχνίδι Limit-Hold'em Poker

Φοιτητής :
Σκλάβος Παναγιώτης 2018030170
Χρυσής Επαμεινώνδας 2018030167

Διδάσκων :
Θρασύβουλος Σπυρόπουλος

Πίνακας Περιεχομένων

1. Εισαγωγή

Όλο και περισσότερες γίνονται στις μέρες μας οι εφαρμογές της ενισχυτικής μάθησης, καθώς παρέχει άμεσες και αποτελεσματικές λύσεις σε μια μεγάλη γκάμα προβλημάτων που συνδέονται με την εύρεση βέλτιστων στρατηγικών και λύσεων. Για τον λόγο αυτό, στα πλαίσια της εργασίας του μαθήματος, υλοποιήθηκαν δύο αρκετά διαδεδομένοι αλγόριθμοι εκ των οποίων ο πρώτος –policy iteration, κατέχει πληροφόρηση για το περιβάλλον του, οπότε εφαρμόζει την θεωρία των MDPs και του Δυναμικού Προγραμματισμού μέσω της συνάρτησης του Bellman, ενώ ο δεύτερος, εργάζεται χωρίς επίγνωση του μοντέλου προκειμένου να επιτύχει σύγκλιση σε βέλτιστη λύση μέσω ειδικής μορφής του Bellman. Στην συνέχεια λοιπόν, πρόκειται να αναλυθούν τα αποτελέσματα της εφαρμογής των άνωθεν αλγορίθμων, σε ένα απλοποιημένο περιβάλλον του παιχνιδιού *Limit Hold'em.* Μέσω του παιχνιδιού αυτού ελέγχεται η αποτελεσματικότητα των αλγορίθμων με βάση την επικράτηση τους ενάντια σε διαφορετικά είδη αντιπάλων.

2. Περιβάλλον Simple Poker Game

Η απλοποιημένη μορφή του παιχνιδιού Poker που αναφέραμε παραπάνω έχει την ακόλουθη μορφή εν συντομία. Η τράπουλα περιέχει 20 $\varphi\acute{\upsilon}\lambda\alpha = 5$ κάρτες$(\mathrm{T}, \mathrm{J}, \mathrm{Q}, \mathrm{K}, \mathrm{A}) * 4\chi\rho\acute{\omega}\mu\alpha\tau\alpha$.Το παιχνίδι διεξάγεται μεταξύ 2 παικτών, ενώ σε καθένα από αυτούς μοιράζεται από μια κάρτα που αποτελεί το «χέρι» του. Κάθε παίκτης υποχρεούται στην αρχή της παρτίδας να εισάγει ένα μικρό ποσό στο παιχνίδι (0.5 blinds). Το ποντάρισμα έπειτα διαχωρίζεται σε δύο γύρους, έναν πριν και έναν μετά το flop (άνοιγμα δύο καρτών στο τραπέζι). Η δομή του πονταρίσματος είναι ίδια και στους δύο γύρους και είναι η ακόλουθη:

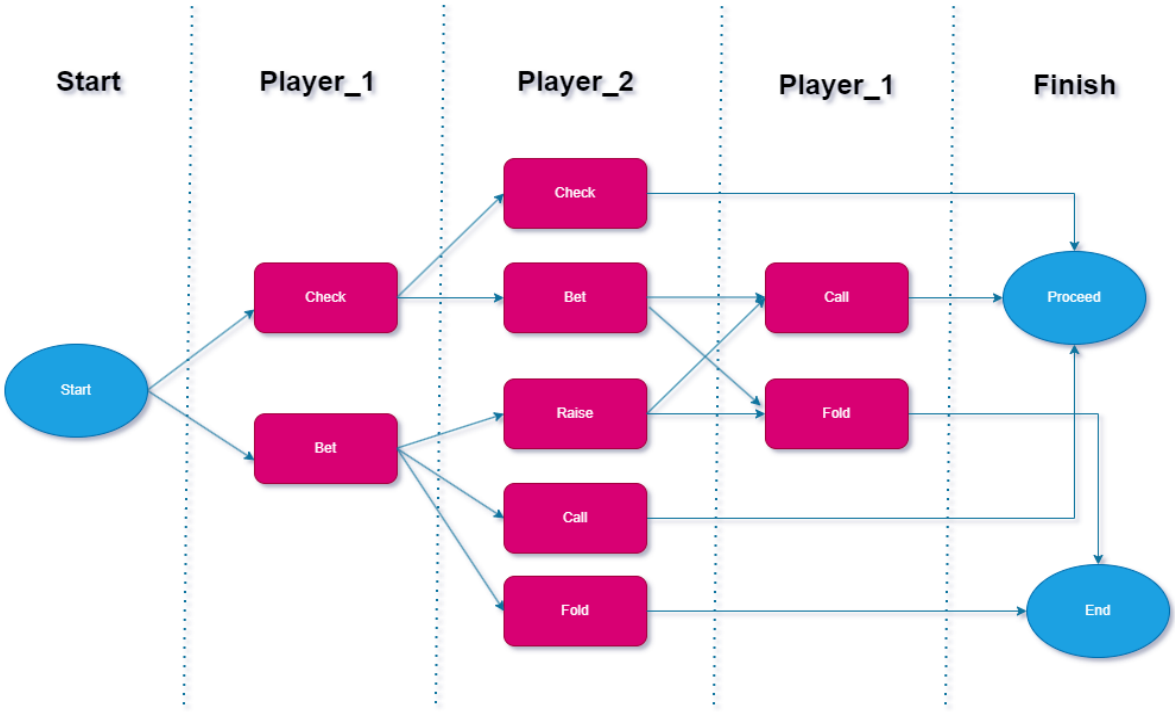


Figure 2.1: Flow-chart αποφάσεων στο παιχνίδι.

Στο σημείο αυτό οφείλει να γίνει μια παρατήρηση σχετικά με το αναγκαστικό αρχικό ποντάρισμα στο παιχνίδι και από τους δύο παίκτες. Το ante όπως αναφέρεται χρησιμοποιείται προκειμένου να αποτραπεί η παθητική συμπεριφορά ώσπου κάποιος παίκτης αποκτήσει πολύ καλές προδιαγραφές για νίκη. Συνεπώς η ύπαρξη αναγκαστικού πονταρίσματος γεννάει την ανάγκη να μεγιστοποιούμε το κέρδος μας ανά γύρο καθώς, διαφορετικά ο παίκτης θα χρεοκοπήσει.

2.1 Υλοποίηση Παιχνιδιού

Για την υλοποίηση του περιβάλλοντος του παιχνιδιού αξιοποιήθηκε σε μεγάλο βαθμό το toolkit RLCard, που έχει αναπτυχθεί από το Data Lab και Texas A&M University. Συγκεκριμένα η υλοποίηση μπορεί να μην εμφανίζει κάποιο Import από την βιβλιοθήκη, ωστόσο βασίζεται σε μεγάλο βαθμό στην υποδομή αυτή. Εν συντομία το παιχνίδι απαρτίζεται από τα ακόλουθα python modules:

Module	***Περιγραφή***
simple_poker_game.py	Περιέχει την κλάση και τις συναρτήσεις που υλοποιούν το παιχνίδι καθώς και πραγματοποιούν τις μεταβάσεις σε αυτό(step).
simple_player.py	Περιέχει την κλάση που περιγράφει έναν παίκτη του παιχνιδιού
simple_round.py	Περιέχει την κλάση που περιγράφει έναν γύρω πονταρίσματος του παιχνιδιού, καθώς και ό,τι μπορεί να συμβεί σε αυτόν
simple_dealer.py	Περιέχει την κλάση του Dealer και μεταξύ άλλων εκεί δομείται η τράπουλα και μοιράζονται τα φύλα
Simple_judger.py	Περιέχει την κλάση του Judger ο οποίος είναι υπεύθυνος να διανείμει τα κέρδη στο τέλος του κάθε γύρου κρίνοντας τον νικητή.
main.py	Εκεί υλοποιείται η παρουσίαση της λειτουργίας του περιβάλλοντος εισάγοντας του επιθυμητούς πράκτορες.
utils.py	Περιέχει πληθώρα συναρτήσεων γενική φύσεως που χρησιμοποιούνται στα υπόλοιπα modules.

Table 2.1 modules που απαρτίζουν το παιχνίδι

2.2 Υλοποίηση Αντιπάλων

Έπειτα δημιουργούνται τέσσερα διαφορετικά είδη αντιπάλων τα οποία χρησιμοποιήθηκαν για τον έλεγχο απόδοσης των αυτοματοποιημένων πρακτόρων. Συγκεκριμένα η λειτουργία τους είναι η ακόλουθη:

Αντίπαλος	***Περιγραφή***
Random Agent	Ο πράκτορας αυτός επιλέγει μια τυχαία κίνηση από τις νόμιμες κινήσεις που του παρέχονται στην τρέχουσα κατάσταση
Threshold Agent: Loose	Pre-flop: bet/raise συμπεριφορά με Q,K,A και check/call με υπόλοιπα Post-flop: bet-raise αν υπάρχει επαφή, αλλιώς check. Μικρή πιθανότητα μπλόφας
Threshold Agent: Tight	Pre-flop: bet/raise με A, bet/call με K, check/call με Q, check/fold διαφ Post-flop: bet/call αν υπάρχει επαφή, αλλιώς fold
Human Agent	Ο χρήστης επιλέγει μία από τις νόμιμες κινήσεις.

Table 2.2 Κατηγορίες Αντιπάλων

2.3 Περιβάλλον Μάθησης

Για την μάθηση της βέλτιστης συμπεριφοράς των πρακτόρων στο περιβάλλον που αναλύθηκε έγιναν ορισμένες σημαντικές επιλογές που αφορούν τον χώρο καταστάσεων, και κατ' επέκτασιν τον τρόπο ορισμού αυτών. Οι επιλογές αυτές αφορούν το χώρο καταστάσεων που γίνεται αντιληπτός από αμφότερους του αλγορίθμους. Επιπλέον, στην περίπτωση της υλοποίησης του Policy Iteration πράκτορα, δεδομένου του ότι ο αλγόριθμος αυτός είναι model-based, δηλαδή, θεωρεί το μοντέλο μεταβάσεων καθώς και τα rewards του MDP γνωστά, δομήθηκε το μοντέλο μεταβάσεων το οποίο, αποτελεί ένα python λεξικό το οποίο εμπερικλείει όλη την πληροφορία του περιβάλλοντός μας. Συγκεκριμένα, το μοντέλο αυτό, περιγράφει για κάθε κατάσταση, τις δυνατές μεταβάσεις (σε επόμενη κατάσταση), με οποιαδήποτε action, καθώς και πληροφορίες για την πιθανότητα της εκάστοτε μετάβασης, το reward της και αν η νέα κατάσταση είναι τερματική

2.3.α Χώρος καταστάσεων

Ο χώρος καταστάσεων που περιγράφει το περιβάλλον του προβλήματος μας επιλέχθηκε να είναι το σύνολο όλων των δυνατών tuples που αποτελούνται από την κάρτα παίκτη και τις κάρτες που έχουν γίνει φανερές στο τραπέζι. Επίσης έχουμε και μια ειδική κατηγορία καταστάσεων που συμβολίζει ότι έχουμε φθάσει σε τερματικό κόμβο. Αξίζει να σημειωθεί ότι εφόσον τα χρώμα δεν αποτελεί κάποια πληροφορία που μας επηρεάζει επιλέχθηκε να μην λαμβάνεται υπ' όψη στην αναπαράσταση καθώς με τον τρόπο αυτό μπορούμε να μειώσουμε στο ένα τέταρτο των πληθυσμό των καταστάσεων. Επιπλέον, εφόσον μας απασχολούν όλοι οι δυνατοί συνδυασμοί προσωπικών και δημόσιων καρτών, δεν λαμβάνεται υπόψιν η διάταξη. Συνεπώς κατά σύβαση, η κατάσταση $('\boldsymbol{A}', ('\boldsymbol{A}', '\boldsymbol{K}'))$ προσμετράται ως $('\boldsymbol{A}', ('K', 'A'))$**,** τοποθετώντας κατά αύξουσα σειρά δυναμικότητας τις δημόσιες κάρτες. Έχουμε λοιπόν συγκεντρωτικά τα εξής είδη καταστάσεων:

Κατάσταση	***Περιγραφή***
$('\boldsymbol{A}', ('\boldsymbol{K}', '\boldsymbol{A}'))$	Hand: 'A' Table: 'K', 'A' Πληθυσμός: $5 \cdot 15 = 75$
$('\boldsymbol{A}', \boldsymbol{None})$	Hand: 'A' Table: όχι κάρτες Πληθυσμός: 5
('Terminal, None)	Αναπαράσταση τερματικής κατάστασης όπου βρίσκουμε νικητή

Table 2.3 Είδη Καταστάσεων

Συγκεντρωτικά λοιπόν ο πληθυσμός του χώρου απαριθμείται ως: $5 + 75 + 1 = 81$ καταστάσεις

2.3.β Μοντέλο Μεταβάσεων

Το μοντέλο μεταβάσεων που χρησιμοποιείται στον Policy Iteration Αλγόριθμο, μεταβάλλεται ανάλογα τον αντίπαλο τον οποίο αντιμετωπίζει. Αυτό συμβαίνει διότι διαθέτουμε πλήρη επίγνωση του περιβάλλοντος καθώς και του είδους του αντιπάλου. Το γεγονός αυτό μας παρέχει την δυνατότητα για κάθε ζεύγος state-action να λαμβάνουμε το σύνολο των δυνατών μεταβάσεων καθώς και τις εξής πληροφορίες: [πιθανότητα μετάβασης, κατάσταση, επιβράβευση, τερματισμός]. Επιγραμματικά λοιπόν θα επεξηγηθεί ο τρόπος με τον οποίο εξάγεται κάθε μια από τις πληροφορίες αυτές.

Πιθανότητα Μετάβασης: Από μια preflop κατάσταση σε μια post flop έχουμε

I. Να εμφανιστεί 2 φορές η κάρτα που κρατάει ο πράκτορας:
$p = P(A \cap B) = P(A) \cdot P(B|A) = \frac{3}{19} \cdot \frac{2}{18} = 1.75\%$. Η πιθανότητα αυτή αποδίδεται σε 1/15 κατάσταση του χώρου(υπάρχουν 15 διαφορετικοί συνδυασμοί καρτών που μπορούν να εμφανιστούν)

II. Να εμφανιστεί 1 φορά η κάρτα που κρατάει ο πράκτορας:
$p = P(A \cup B) = P(A) + P(B) - P(A \cap B) = \frac{3}{19} + \frac{3}{19} - \frac{6}{19*18} = 29.82\%$. Η πιθανότητα αυτή μοιράζεται σε 4 διαφορετικές και ισοπίθανες καταστάσεις

III. Να μην εμφανιστεί η κάρτα: : $p = 100\% - 29.82\% - 1.75\%$=68.43%. Η πιθανότητα αυτή διανέμεται σε 10 ισοπίθανες καταστάσεις.

Επιβράβευση: ανάλογα με τον αντίπαλο έχει δομηθεί ένα διαφορετικό reward model το οποίο είναι σχεδιασμένο ώστε να παρέχει μεγαλύτερες επιβραβεύσεις σε actions τα οποία, με βάση την στρατηγική του δεδομένου του αντιπάλου, είναι περισσότερο πιθανόν να οδηγήσουν τον πράκτορα σε νίκη. Η υποκείμενη λογική πίσω από τα reward model του εκάστοτε αντιπάλου θα αναλυθεί σε επόμενο κομμάτι.

Τερματισμός: Αποτελεί μια Boolean μεταβλητή που γίνεται αληθής όταν μεταβαίνουμε σε τερματική κατάσταση (Terminal state).

3. Model-Based: Policy Iteration Πράκτορας

Στο τμήμα αυτό πρόκειται να σχολιαστούν, αφενός οι επιλογές που πραγματοποιήθηκαν σχετικά με τα διαφορετικά μοντέλα μεταβάσεων που αφορούν τα διαφορετικά είδη αντιπάλων, και αφετέρου οι επιλογές σχετικά με τις υπερπαραμέτρους του αλγορίθμου και τα πειράματα που παρουσιάζουν την λειτουργία του.

3.1 Μοντέλα Μεταβάσεων Αντιπάλων

Το περιβάλλον σε κάθε περίπτωση παραμένει σταθερό, όπως γίνεται εύκολα αντιληπτό. Αυτό έχει ως άμεσο αντίκτυπο να μην μεταβάλλονται τα πεδία που αφορούν τις πιθανότητες μετάβασης καθώς και τις ιδιότητες των καταστάσεων. Το πεδίο το οποίο αναπροσαρμόζεται λοιπόν, ανάλογα τον αντίπαλο είναι εκείνο που αφορά το δοθέν reward για την πραγμάτωση μια συγκεκριμένης μετάβασης. Είναι λογικό εφόσον κάθε αντίπαλος ακολουθεί διαφορετική στρατηγική, να προσαρμόζονται οι ανταμοιβές με τέτοιο τρόπο ώστε να επιβραβεύουν κινήσεις που αυξάνουν την πιθανότητα νίκης του υπό μελέτη πράκτορα, εκμεταλλευόμενοι τις γνώσεις για τις προδιαθέσεις του αντιπάλου.

Συγκεκριμένα, σε πρώτη φάση θα αναλυθεί η στρατηγική ενάντια στον Random Agent. Με δεδομένο το γεγονός ότι δεν εφαρμόζεται κάποια έγκυρη συλλογιστική στην λήψη αποφάσεων για τον αλγόριθμο αυτό, το μοντέλο σχεδιάζεται ώστε να επιβραβεύει περισσότερο κινήσεις πονταρίσματος. Αυτό συμβαίνει διότι αν ο αντίπαλος τίθεται 2 φορές σε ισάριθμους γύρους πονταρίσματος, να αποφασίσει μεταξύ τριών επιλογών (fold, call, raise) εκ των οποίων η μία σημαίνει αυτόματη νίκη για τον on policy πράκτορά, έχουμε: $\mathrm{p} = \frac{1}{3} + \frac{1}{3} - \frac{1}{3} \cdot \frac{1}{3} = 5/9$ πιθανότητα νίκης χωρίς να επέλθει κατάσταση showdown. Με άλλα λόγια 5 στις 9 φορές ο πράκτορας νικάει άμεσα. Από τις υπόλοιπες 4 φορές ωστόσο που καταλήγει σε showdown η πιθανότητα να νικάει παραμένει 50%. Άρα, νίκη σε δύο από τις υπολειπόμενες τέσσερις. Αυτό έχει ως αποτέλεσμα σε 9 παιχνίδια ο πράκτορας να αποβαίνει νικητής στα 7. Συνεπώς πολύ γρήγορα καταλήγει σε συνολική νίκη.

Έπειτα, ενάντια στον threshold loose, η στρατηγική που έχει επιλεχθεί επιβραβεύει σε μεγαλύτερο βαθμό προσεκτικό τρόπο παιξίματος, αποφεύγοντας να τοποθετούνται χρήματα στο ποτ χωρίς καλές προϋποθέσεις νίκης. Αντίθετα όταν οι πιθανότητες νίκης του πράκτορα είναι ικανοποιητικές επιβραβεύονται περισσότερο actions τα οποία μεγιστοποιούν τα χρήματα που τοποθετούνται στο παιχνίδι. Η ταχύτητα που επιτυγχάνεται η νίκη είναι αρκετά καλή και στην περίπτωση αυτή.

Τέλος, η μεθοδολογία που επιβραβεύεται περισσότερο ενάντια στον tight threshold, παροτρύνει τον πράκτορα να παίζει περισσότερα χέρια. Με τον τρόπο αυτό αυξάνει την πιθανότητά του να βρεθεί σε επικερδείς καταστάσεις. Αν ο αντίπαλος δείξει μετά το φλοπ κάποια επιθετική πρόθεση, ο πράκτορας επιβραβεύεται ώστε να κάνει fold και να χάνει ελάχιστα χρήματα, διαφορετικά αν ο αντίπαλος δεν γίνει επιθετικός, ο πράκτορας έχει κίνητρο να μπλοφάρει και να κλέψει την νίκη. Η μεθοδολογία αυτή είναι πιο αργή ως προς το πότε επιτυγχάνεται η νίκη, αλλά έχει σταθερά αποτελέσματα

3.2 Πειράματα και παρατηρήσεις Policy Iteration

Για τους πειραματισμούς που αφορούν την λειτουργία του Policy Iteration έχει επιλεχθεί μια πολύ κλασσική ανάθεση παραμέτρων: ως όριο σύγκλισης τοποθετείται: $epsilon = 1^{-10}$, συνεπώς επιχειρούμε να επιτύχουμε convergence σε βαθμό που πρακτικά δεν μεταβάλλονται πλέον το Value function από τα νέα policies. Έπειτα discount factor έχει τεθεί $gamma = 0.95$. Η επιλογή αυτή έγινε, καθώς μας ενδιαφέρει να λαμβάνουμε υπόψιν τις μελλοντικές καταστάσεις και reward που απορρέουν από μια κίνηση μακροπρόθεσμα. Συνεπώς το gamma τοποθετείται κοντά στην μονάδα. Ωστόσο από την θεωρία έχουμε δει ότι για το gamma<1 έχει ευκολότερη συμπεριφορά ως προς την σύγκλιση.

Οι τιμές αυτές των παραμέτρων επέτυχαν τον στόχο ταχείας σύγκλισης ενώ ταυτόχρονα οι αποφάσεις που λαμβάνονται είναι βέλτιστες με βάση το εκάστοτε μοντέλο μεταβάσεων που παρατίθεται. Τα αποτελέσματα που ακολουθούν αφορούν έναν πράκτορα του οποίου το policy έχει ήδη συγκλίνει, συνεπώς δεν παρατηρούμε κάποιο περιβάλλον μάθησης, αλλά πειραματισμού. Η σύγκλιση στην βέλτιστη λύση έχει επιτευχθεί σε μόλις 3 γύρους policy

iteration, γεγονός που οφείλεται, αφενός στον μικρό χώρο καταστάσεων, και αφετέρου, στις αρκετά διαφορετικές τιμές που έχουν δοθεί στο reward model οδηγώντας ταχύτερα σε έγκυρα αποτελέσματα.

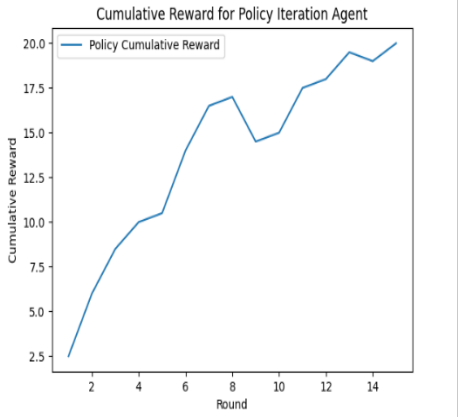
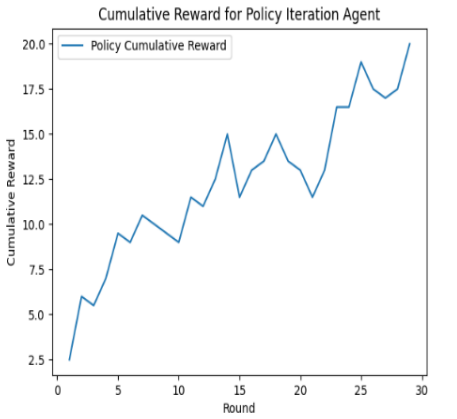
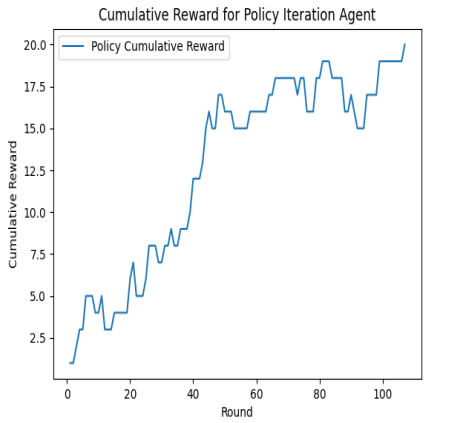
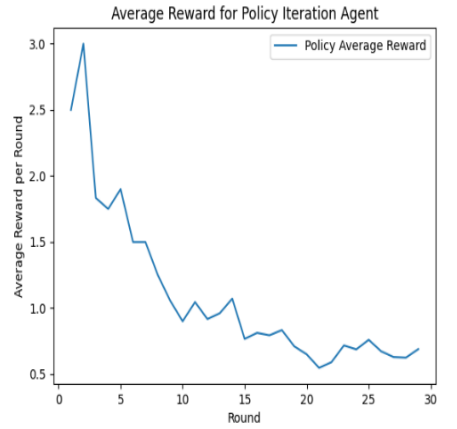
Random Agent	Threshold Loose Agent	Threshold Tight Agent
		
		

Table 3.1: Πίνακας με γραφικές απεικονίσεις του αθροιστικού reward στην πρώτη γραμμή και του μέσου στην δεύτερη, ανά γύρο

Όπως γίνεται αντιληπτό το policy που καταλήγει ο πράκτορας να εφαρμόζει μετά την χρήση του policy iteration αλγορίθμου οδηγεί σε επικερδής αποφάσεις που οδηγούν τελικά στην νίκη. Στα παραδείγματα που παρατηρούμε οι παίκτες διαγωνίστηκαν σε ένα παιχνίδι με 20 μάρκες ο καθένας στην αρχή, με τερματική συνθήκη την χρεοκοπία του αντιπάλου.

Πολυάριθμες είναι οι παρατηρήσεις που μπορούν να αναφερθούν από τον παραπάνω διαγωνισμό. Αρχικά, από την πρώτη γραμμή του πίνακα, από τος σχήμα της γραφικής, γίνεται αντιληπτό ότι δεν επιτυγχάνεται νίκη του αντιπάλου σε όλους τους γύρους. Γεγονός που είναι αναμενόμενο καθώς παρόλο που η στρατηγική του αντιπάλου είναι γνωστή, το Poker παραμένει παιχνίδι πιθανότητας καθώς και επίσης η κάρτα του αντιπάλου δεν είναι γνωστή στο πράκτορα. Συνεπώς ακόμη και με ορθή πολιτική απόφασης είναι λογικό να οδηγείται σε ορισμένες αστοχίες.

Επιπλέον, όπως είναι αναμενόμενο η νίκη ενάντια στον Random αντίπαλο επέρχεται πολύ γρήγορα, ενάντια στο Loose Threshold ικανοποιητικά, και εναντίον του tight με χαμηλότερο ρυθμό. Τα αποτελέσματα αυτά παραμένουν σταθερό όσα πειράματα κι αν επιχειρήσουμε, καθώς ο πρώτος μπορεί να νικηθεί πολύ εύκολα αν γίνει εκμετάλλευση των

αδυναμιών του, ο δεύτερος, ποντάρει πιο εύκολα τα χρήματά του, ακόμη κι αν δεν έχει τις καλύτερες πιθανότητες νίκης, ενώ ο τρίτος είναι ιδιαίτερα προσεκτικός και παρόλο που εξαναγκάζεται σε πολλές ήττες λόγω δικής του αστοχίας, και μπλόφας από τον Policy Iteration πράκτορα, η ζημία του δεν είναι αρκετά μεγάλη ώστε να επιφέρει ταχεία χρεοκοπία.

Τέλος, σημαντικό είναι να αναφερθεί ότι ο αλγόριθμος καταλήγει σε βέλτιστη στρατηγική, ωστόσο βέλτιστη όπως την έχει θεωρηθεί από την δημιουργία του μοντέλου μεταβάσεων και δοθέντων ανταμοιβών. Με άλλα λόγια, η βελτιστότητα στην περίπτωσή μας, σημαίνει απόλυτη σύγκλιση στην καλύτερη στρατηγική που απορρέει από το παρόν μοντέλο και όχι καθολικά βέλτιστη μεταξύ όλων των στρατηγικών. Ένα μοντέλο μετάβασης που θα οδηγεί σε τέτοιου είδους στρατηγικής θα απαιτούσε εξαιρετική γνώση και μελέτη των κανόνων του παιχνιδιού και μαθηματική ανάλυση. Η επιβεβαίωση λοιπόν της βελτιστότητας των αποφάσεων του πράκτορα απορρέει από το γεγονός ότι καταληκτικά, εκτελεί τις αναμενόμενες πράξεις που υπαγορεύονται από τον πίνακα μεταβάσεων, δηλαδή ο πράκτορας εφαρμόζει την συμπεριφορά την οποίο κλήθηκε να μάθει.

4. Model-Free: Q-Learning Πράκτορας

Ο αλγόριθμος αυτός, δεν έχει πληροφόρηση σχετικά με τον αντίπαλο και το περιβάλλον στο οποία εφαρμόζεται. Συνεπώς στο μέρος αυτό υλοποιείται μια αγνωστική προσέγγιση μέσω της οποίας επιχειρείται να επιτευχθεί μάθηση βέλτιστης στρατηγικής μέσω μάθησης των action values (Q(s,a)) και τελικά σύγκλισης στα βέλτιστα που παρέχουν ορθό κανόνα απόφασης.

Ο πράκτορας που υλοποιεί τον αλγόριθμο, προτού ξεκινήσει να διαγωνίζεται με τον αντίπαλό του τρέχει έναν μεγάλο αριθμό προσομοιώσεων, ώσπου τελικά να επιτύχει σύγκλιση στα $Q^*(s,a)$- βέλτιστα. Έπειτα έχοντας επιτύχει σύγκλιση έρχεται αντιμέτωπος με τον αντίπαλο εφαρμόζοντας την γνώση που συγκέντρωσε.

4.1 Μάθηση των Q(s,a) - Training Phase

4.1 Ψευδοκώδικας Q-Learning

Algorithm paramet

Initialize $Q(s,a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal,\cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
    $S \leftarrow S'$
  until S is terminal

Ο γενικός αλγόριθμος Q- Learning που παρατηρούμε αριστερά αποτελεί τον κορμό της φάσης της εκπαίδευσης. Για τις συνθήκες του δικού μας προβλήματος ωστόσο έχουν πραγματοποιηθεί μερικές τροποποιήσεις.

Συγκεκριμένα, στο περιβάλλον μελέτης μας έχουμε 2 αντιπάλους να λαμβάνουν action, συνεπώς αντί να λαμβάνεται απλά ένα action ελέγχεται ποιος παίκτης παίζει. Αν πρόκειται για τον Q- Learning Agent επιλέγεται action με βάση epsilon greedy πολιτική. Διαφορετικά αν παίζει ο αντίπαλος επιλέγει την κίνησή του με βάση την δική του στρατηγική. Συνεπώς στον κόσμο του παιχνιδιού κατηγοριοποιούνται 2 στιγμές που μπορεί να γίνει update του Q table.

Η πρώτη είναι, αν μετά από πράξη του Q-Learning agent πραγματοποιήσει δική του ο αντίπαλος. Με άλλα λόγια το ένα step στο παιχνίδι συνεπάγεται τόσο δική μας κίνηση όσο και την απάντηση του αντιπάλου. Κατά τον τρόπο αυτό το αντίκτυπο της πράξης (reward) αποδίδεται στην αρχική κίνηση του πράκτορα. Συγκεκριμένα, αρχικό state θεωρείται το state του πράκτορα και next state αναγνωρίζεται η κατάσταση που θα επέλθουμε μετά την κίνηση του αντιπάλου.

4.1.2 Παράδειγμα Q-table

Started Training Q-Learning Agent
Q-learning converged in 99708 episodes

State	Action	Q-Value
('T', None)	fold	58.71
('T', None)	check	60.42
('T', None)	bet	60.39
('T', None)	call	58.93
('T', None)	raise	59.58
('T', ('T', 'T'))	fold	6.81
('T', ('T', 'T'))	check	13.80
('T', ('T', 'T'))	bet	40.03
('T', ('T', 'T'))	call	25.47
('T', ('T', 'T'))	raise	31.76
('T', ('T', 'J'))	fold	32.43
('T', ('T', 'J'))	check	67.15
('T', ('T', 'J'))	bet	77.86
('T', ('T', 'J'))	call	73.40
('T', ('T', 'J'))	raise	73.64
('T', ('T', 'Q'))	fold	31.87
('T', ('T', 'Q'))	check	59.21
('T', ('T', 'Q'))	bet	69.60
('T', ('T', 'Q'))	call	67.69
('T', ('T', 'Q'))	raise	65.64
('T', ('T', 'K'))	fold	38.81
('T', ('T', 'K'))	check	63.10
('T', ('T', 'K'))	bet	71.55
('T', ('T', 'K'))	call	66.84
('T', ('T', 'K'))	raise	66.47
('T', ('T', 'A'))	fold	42.84
('T', ('T', 'A'))	check	60.83
('T', ('T', 'A'))	bet	68.19
('T', ('T', 'A'))	call	64.88
('T', ('T', 'A'))	raise	64.41
('T', ('J', 'J'))	fold	6.69
('T', ('J', 'J'))	check	25.03
('T', ('J', 'J'))	bet	48.30
('T', ('J', 'J'))	call	37.56
('T', ('J', 'J'))	raise	39.66
('T', ('J', 'Q'))	fold	42.02
('T', ('J', 'Q'))	check	55.91
('T', ('J', 'Q'))	bet	64.46
('T', ('J', 'Q'))	call	58.93
('T', ('J', 'Q'))	raise	59.36
('T', ('J', 'K'))	fold	42.45
('T', ('J', 'K'))	check	54.78

Η δεύτερη περίπτωση είναι όταν ο ίδιος ο πράκτορας κλείσει την δράση μια φάσης πονταρίσματος, συνεπώς με το action του οδηγείται σε να νέα κατάσταση επόμενου γύρου(ή τερματική). Κατά τον τρόπο αυτό ο πίνακας ανανεώνεται σε κάθε step μέσα στον κόσμο του παιχνιδιού, με το step να περιλαμβάνει το αντίκτυπο των κινήσεων και των 2 αντιπάλων.

Τελευταία διαφορά αποτελεί το γεγονός ότι η κατάσταση του training στην παρούσα υλοποίηση δεν ολοκληρώνεται σε σταθερό αριθμό επεισοδίων, αλλά όταν γίνει τελικά converge των action values. Ο έλεγχος αυτός επιτυγχάνεται συγκρίνοντας την μέγιστη διαφοροποίηση στον πίνακα των Q με ένα threshold (default = 0.0001)

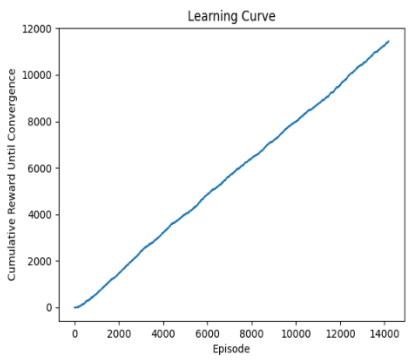
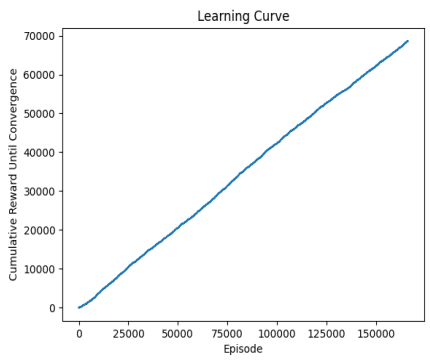
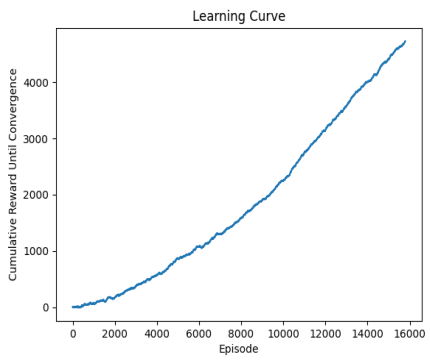
Καταλήγοντας, οφείλει να γίνει αναφορά στο reward system το οποίο είναι αρκετά διαφορετικό από εκείνο που μελετήθηκε κατά το Policy Iteration. Συγκεκριμένα, στην παρούσα προσέγγιση το reward που αποδίδεται είναι 0 αν βρισκόμαστε σε κάποια ενδιάμεση κατάσταση, αλλιώς τα κέρδη/χαμένα αν βρισκόμαστε σε τερματική.

4.2 Πειράματα και Παρατηρήσεις Q-Learning

Προκειμένου να επέλθει σύγκλιση στο βέλτιστο έχει γίνει η ακόλουθη επιλογή παραμέτρων: $epsilon = (episode)^{-1/4}$, για πειράματα που συγκλίνουν σε τάξη μεγέθους > 20,000 επεισοδίων, ενω για $epsilon = (episode)^{-1/5}$, για πειράματα που συγκλίνουν σε τάξη μεγέθους > 100,000

Έπειτα το $learning\_rate = 0.1$: η τιμή αυτή δίνεται καθώς στο μέγεθος του προβλήματος μας η σύγκλιση επιτυγχάνεται σε σύντομο χρόνο, οπότε προτιμήθηκε ένα σχετικά μικρό βήμα μάθησης προκειμένου να μην γίνει overshoot των βέλτιστων Q values.

Τέλος $gamma = 0.95$, καθώς η πλειοψηφία τον Q-Values ανανεώνονται από μελλοντικές στιγμές παρά το εγγύς reward, όποτε προτιμάται να δίνεται μεγάλη βαρύτητα στο μέλλον. Ωστόσο επιλέχθηκε τιμή μικρότερη της μονάδας προκειμένου να διευκολυνθεί η σύγκλιση

Random Agent	**Threshold Loose Agent**	**Threshold Tight Agent**
		

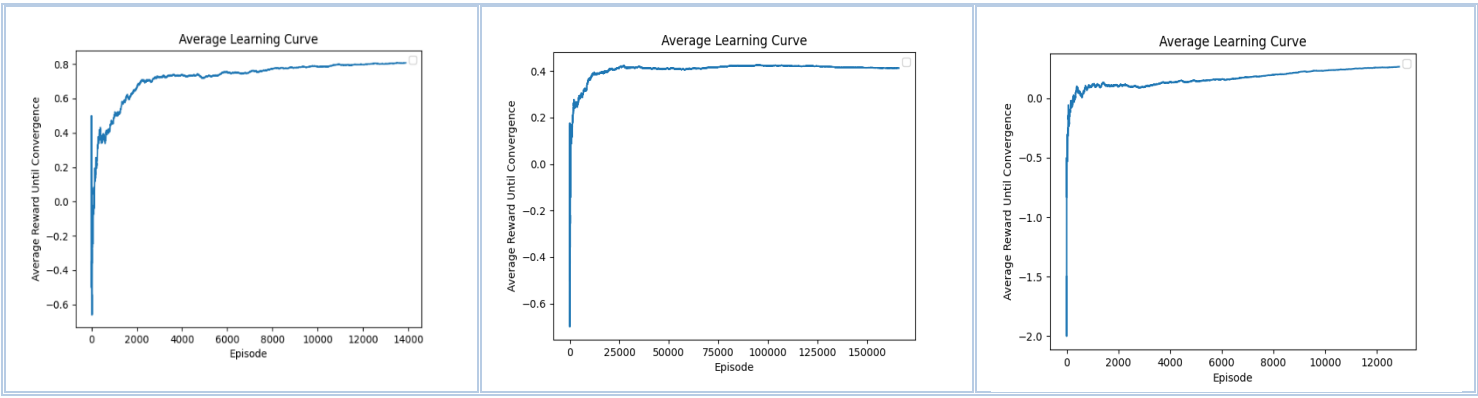


Table 4.1: Πίνακας με γραφικές απεικονίσεις του αθροιστικού reward στην πρώτη γραμμή και του μέσου στην δεύτερη, ανά επεισόδιο

4.3 Χρόνος Σύγκλισης και Scalability

Ο χρόνος για να επιτευχθεί σύγκλιση σε κάθε μια από τις παραπάνω περιπτώσεις ποικίλει αναλόγως την επιλογή τον παραμέτρων. Συγκεκριμένα για τον random απαιτείται χρόνος περίπου 5 δευτερολέπτων, για τον tight μόλις 3 δευτερόλεπτα ενώ για τον loose αναλόγως την επιλογή παραμέτρων ποικίλει από 30 δευτερόλεπτα έως ένα λεπτό. Προφανές λοιπόν γίνεται, ότι παρόλο που στο μικρό σχετικά χώρο καταστάσεων η απόδοση του Q-Learning είναι ικανοποιητική, η μελέτη μεγαλύτερων προβλημάτων θέτει μεγάλο υπολογιστικό φόρτο και απαιτεί σημαντικό χρόνο. Ακόμη και στον πειραματισμό με διαφορετικές παραμέτρους, στον χώρο καταστάσεων της άσκησής, μπορούμε να παρατηρήσουμε ότι η διαδικασία της σύγκλισης μπορεί με μεγάλη ευκολία να γίνει σημαντικά πιο αργή.

4 Σύγκριση Αλγορίθμων

Οι διαφορές που απορρέουν από τα δομικά χαρακτηριστικά των αλγορίθμων δεν θα σχολιαστούν καθώς έχουν αναφερθεί εκτενώς σε προηγούμενα τμήματα. Ωστόσο αξίζει να γίνει μια άμεση σύγκριση της απόδοσης των 2 στο περιβάλλον του Simple Poker. Συγκεκριμένα έχουμε:

Ενδεικτικό Average Reward (μάρκες ανά χέρι)			
	Random Agent	*Threshold Loose Agent*	*Threshold Tight Agent*
Policy Iteration Agent	2.2	0.44	0.24
Q-Learning Agent	0.81	1.17	0.86

Τα αποτελέσματα αυτά είναι ενδεικτικά και διαφέρουν από πείραμα σε πείραμα ωστόσο δεν απέχουν και πολύ από την καταληκτική εικόνα. Αναλυτικότερα, το μοντέλο που έχει υλοποιηθεί στον Policy Iteration Agent για τον Random αλγόριθμο είναι εξαιρετικά αποδοτικό και στην πλειοψηφία των περιπτώσεων έχει καλύτερα αποτελέσματα από εκείνα του Q-Learning, γεγονός που σημαίνει ότι συγκλίνει σε υποβέλτιστη στρατηγική. Από την άλλη ωστόσο, όσον αφορά τους δύο threshold αντιπάλους, ο Q-Learning έχει εντυπωσιακά αποτελέσματα τα οποίο είναι στην πλειοψηφία τους καλύτερα από του Policy Iteration. Το γεγονός αυτό προφανώς και αποδίδεται στην μη βελτιστότητα της στρατηγικής που προωθούν τα υλοποιημένα μοντέλα ανταμοιβών για τον on Policy Agent. Συνοψίζοντας ωστόσο, τόσο ο Policy όσο και Q-Learning Agents έχουν σταθερή απόδοση που σημαίνει ότι ανακηρύσσονται νικητές σχεδόν πάντοτε, με ελάχιστες εξαιρέσεις.

5 Επίλογος

Συνοψίζοντας, υλοποιήθηκαν δύο διαφορετικά μοντέλα ενισχυτικής μάθησης τα οποία με τα δικά του ιδιαίτερα χαρακτηριστικά, μπόρεσαν να αποφέρουν αποδοτικές και βέλτιστες λύσεις στο περιβάλλον του Simple Poker Game.

Ανοικτό για το μέλλον παραμένει το μέτωπο της βελτιστοποίησης των παραμέτρων ώστε να μπορέσει να επιτευχθεί ο συγκερασμός βέλτιστης απόδοσης και σύγκλιση στην optimal λύση. Επίσης, ενδιαφέρον έχει η μελέτη της λειτουργικότητας και την προσαρμογής των πρακτόρων ώστε να μπορέσουν να ανταποκριθούν και σε δυσκολότερες παραλλαγές του συστήματος, μέχρι τον βαθμό που αυτό είναι εφικτό.