ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μοντελοποίηση και βελτίωση της ανθρώπινης ικανότητας σε παιχνίδια στρατηγικής

Ντούτση Ειρήνη Α.Μ. 1934

Υπεύθυνος Καθηγητής: Ελευθέριος Κυρούσης, Καθηγητής
Επιβλέπων: Δημήτρης Καλλές, Διδάκτωρ

Πάτρα 2001

Στην οικογένειά μου
και
στους φίλους μου

Ευχαριστίες

Ευχαριστώ πολύ όλους όσους με βοήθησαν και με στήριξαν με οποιοδήποτε τρόπο κατά τη διάρκεια της διπλωματικής. Πιο συγκεκριμένα, ευχαριστώ τον κύριο Δημήτρη Καλλέ για τις συμβουλές του και τις κατευθύνσεις που μου έδωσε είτε μέσα από τις επιστημονικές εργασίες που μου πρότεινε είτε μέσα από τις συζητήσεις μας.

Ευχαριστώ επίσης τον κύριο Ελευθέριο Κυρούση για την εμπιστοσύνη που μου έδειξε κατά τη διάρκεια της διπλωματικής και για τη διάθεση του εργαστηρίου του τομέα για τα περάματα. Πολλά ευχαριστώ και στον κύριο Ηλία Σταυρόπουλο για τη βοήθειά του στο εργαστήριο.

Ευχαριστώ πολύ τον Παναγιώτη Κανελλόπουλο για τη βοήθεια και τη διαθεσιμότητά του καθ' όλη τη διάρκεια της διπλωματικής. Επίσης ευχαριστώ τον Αθανάσιο Παπαγγελή για τις εύστοχες παρατηρήσεις του και τις συμβουλές του.

Τέλος, ευχαριστώ πολύ του φίλους μου για το ενδιαφέρον τους και τη διάθεσή τους να βοηθήσουν. Κυρίως ευχαριστώ το Νικόλαο Μήτσου για την ουσιαστική του βοήθεια και τον Κώστα Γρατσία για το ενδιαφέρον του καθ' όλη τη διάρκεια της διπλωματικής.

Σκοπός της διπλωματικής

Ο βασικός σκοπός κατά τον σχεδιασμό της διπλωματικής ήταν η μελέτη του τρόπου με τον οποίο μπορεί να μοντελοποιηθεί η συμπεριφορά ενός παίκτη ώστε να εξαχθούν συμπεράσματα για τις δυνατότητες και τις αδυναμίες του αλλά και να χαραχθεί μια στρατηγική για την βελτίωση του. Παράλληλος στόχος ήταν η υλοποίηση της μοντελοποίησης έτσι όπως αυτή θα προέκυπτε από την προαναφερθείσα θεωρητική μελέτη.

Η παρούσα διπλωματική στηρίζεται σε ένα ήδη υπάρχουν παιχνίδι στρατηγικής που μελετήθηκε για πρώτη φορά στα πλαίσια της διπλωματικής εργασίας με τίτλο «*Ανακάλυψη κανόνων για παιχνίδια στρατηγικής, Διπλωματική εργασία ΤΜΗΥΠ, 2000*», από τον κύριο Παναγιώτη Παπαϊωάννου. Όσον αφορά στη χρήση Ενισχυτικής Μάθησης στο παιχνίδι, αυτή είχε ήδη αναπτυχθεί στα πλαίσια της ερευνητικής δραστηριότητας της Ερευνητικής Μονάδας 3 (ΕΜ3) του Ινστιτούτου Τεχνολογίας Υπολογιστών (ΙΤΥ) από τους κυρίους Δημήτρη Καλλέ και Παναγιώτη Κανελλόπουλο. Η έρευνα αυτή οδήγησε σε μια επιστημονική δημοσίευση [Kalles & Kanellopoulos 2001] η οποία και χρησιμοποιήθηκε ως πηγή μελέτης για την εκπόνηση της διπλωματικής. Εδώ θα πρέπει να ευχαριστήσω για μία ακόμη φορά τους κυρίους Καλλέ και Κανελλόπουλο για το χρόνο που μου αφιέρωσαν και τη βοήθειά τους.

Πέραν των αρχικών απαιτήσεων στην παρούσα διπλωματική σχεδιάστηκε και υλοποιήθηκε και το περιβάλλον αλληλεπίδρασης με το χρήστη.

Πρόλογος

Η αξία της Ενισχυτικής Μάθησης *(RL)* έχει αρχίσει να αναγνωρίζεται από πολλούς ερευνητές. Η σοβαρή θεωρητική θεμελίωση του τομέα παρακινεί όλο και περισσότερους επιστήμονες να ασχοληθούν προκειμένου να κατασκευάσουν συστήματα που θα δουλεύουν σε πραγματικές συνθήκες και να βελτιώσουν τα κλασικά συστήματα ελέγχου.

Το γεγονός αυτό δεν είναι υπερβολικό αν αναλογιστεί κανείς τις τεράστιες δυνατότητες του τομέα, πολλές απ΄ τις οποίες δεν έχουν διερευνηθεί ακόμη πλήρως. Και μόνο το ότι ένα σύστημα μπορεί να μαθαίνει να επιλύει κάποιο πρόβλημα πειραματιζόμενο με το περιβάλλον μάθησης είναι εντυπωσιακό. Αυτή η γενικότητα της μάθησης επιτρέπει την εφαρμογή της ίδιας μεθόδου μάθησης και σε άλλα προβλήματα. «Το πρόβλημα προσαρμόζεται στη λύση» όπως αναφέρει χαρακτηριστικά ο Sutton, ένας από τους κορυφαίους επιστήμονες του τομέα.

Η παρούσα διπλωματική βασίζεται σε ένα πρωτότυπο παιχνίδι στρατηγικής που χρησιμοποιεί τεχνικές Ενισχυτικής Μάθησης *(Reinforcement Learning)* για τη βελτίωσή του. Στα πλαίσια της διπλωματικής μελετήσαμε τον τρόπο με τον οποίο μπορεί να μοντελοποιηθεί η συμπεριφορά ενός παίκτη ώστε να εξαχθούν συμπεράσματα για τις δυνατότητες και τις αδυναμίες του αλλά και να χαραχθεί μια στρατηγική για την βελτίωση των ικανοτήτων του.

Ποιο συγκεκριμένα, μελετήσαμε τους πιθανούς τρόπους μοντελοποίησης / σύγκρισης μεταξύ πολύπλοκων οντοτήτων (ανθρώπινη συμπεριφορά/ διαφορετικές διαμορφώσεις της "σκακιέρας") και υλοποιήσαμε ένα μηχανισμό μοντελοποίησης των παικτών .

Το παρόν σύγγραμμα αποτελείται από 8 κεφάλαια. Το πρώτο κεφάλαιο είναι μια εισαγωγή στο παιχνίδι: από ποια στοιχεία αποτελείται, πως παίζεται, ποιοι είναι οι κανόνες κ.α. Το δεύτερο, τρίτο και τέταρτο κεφάλαιο αναφέρονται στην Ενισχυτική Μάθηση: τι είναι, που χρησιμοποιείται, ποιες είναι οι βασικές της έννοιες, τι είναι η μάθηση χρονικών διαφορών, τι είναι τα ίχνη καταλληλότητας κ.α. Το πέμπτο κεφάλαιο αναφέρεται στα Νευρωνικά Δίκτυα: τι είναι, πως δουλεύουν, ποιους αλγορίθμους χρησιμοποιούν κ.α. Το έκτο κεφάλαιο αναφέρεται στη μοντελοποίηση: τι σημαίνει, τι προσφέρει, ποιες τεχνικές υπάρχουν, πως συνδυάζεται με τη Μηχανική Μάθηση κ.α. Στο έβδομο κεφάλαιο αναλύεται πως η προαναφερθείσα θεωρία εφαρμόζεται στο παιχνίδι μας και περιγράφεται η αρχιτεκτονική του συστήματος και το γραφικό του περιβάλλον. Στο όγδοο κεφάλαιο αναφέρονται κάποια βασικά συμπεράσματα έτσι όπως προέκυψαν κατά τη διάρκεια της διπλωματικής μαζί με προτεινόμενες μελλοντικές κινήσεις για τη βελτίωση του παιχνιδιού. Ακολουθεί η βιβλιογραφία, το παράρτημα με κάποιους βασικούς όρους και τις επεξηγήσεις τους και τέλος το ευρετήριο σχημάτων και πινάκων.

Ειρήνη Ντούτση

Πάτρα 2001

Περιεχόμενα

1. Περιγραφή παιχνιδιού

1.1. Γενικά

Για τους σκοπούς της διπλωματικής μελετήθηκε ένα πρωτότυπο παιχνίδι στρατηγικής [Kalles & Kanellopoulos 2001]. Το παιχνίδι παίζεται με δύο αντιπάλους σε μία τετραγωνική σκακιέρα. Κάθε αντίπαλος ξεκινάει από μία τετραγωνική βάση (υπάρχουν δύο τέτοιες στη σκακιέρα, μία πάνω δεξιά και μία κάτω αριστερά) και έχει στη διάθεση του ένα αριθμό πιονιών. Τόσο το μέγεθος της βάσης, όσο και το πλήθος των πιονιών είναι κοινά για τους δύο αντιπάλους. Στόχος του κάθε παίκτη είναι να καταλάβει την αντίπαλη βάση προστατεύοντας συγχρόνως τη δικιά του από τα πιόνια του αντιπάλου. Ο παίκτης που θα καταφέρει να βάλει πρώτος κάποιο πιόνι του στη βάση του αντιπάλου θεωρείται νικητής. Αν κάποιος παίκτης ξεμείνει από πιόνια, ο αντίπαλος του θεωρείται αυτόματα νικητής.

1.2. Συστατικά του παιχνιδιού

Η σκακιέρα

Το παιχνίδι διαδραματίζεται πάνω σε μία τετραγωνική σκακιέρα διαστάσεων *n*n.* Το μέγεθος της σκακιέρας είναι μεταβλητό και καθορίζεται από το χρήστη. Κατά τη διάρκεια του παιχνιδιού, οποιοδήποτε τετραγωνάκι της σκακιέρας μπορεί να είναι κενό ή να περιέχει κάποιο πιόνι.

Οι βάσεις

Υπάρχουν δύο βάσεις διαστάσεων *a*a* μία για κάθε παίκτη. Στην αρχή του παιχνιδιού, κάθε βάση περιέχει τα πιόνια του αντίστοιχου παίκτη. Οι δύο παίκτες ανταγωνίζονται για το ποιος θα βάλει πρώτος κάποιο πιόνι του στην αντίπαλη βάση κερδίζοντας έτσι το παιχνίδι.

Τα πιόνια

Χρησιμοποιούνται για την διεξαγωγή του παιχνιδιού. Στην αρχή του παιχνιδιού κάθε παίκτης έχει στην κατοχή του *β* πιόνια λευκού ή μαύρου χρώματος.

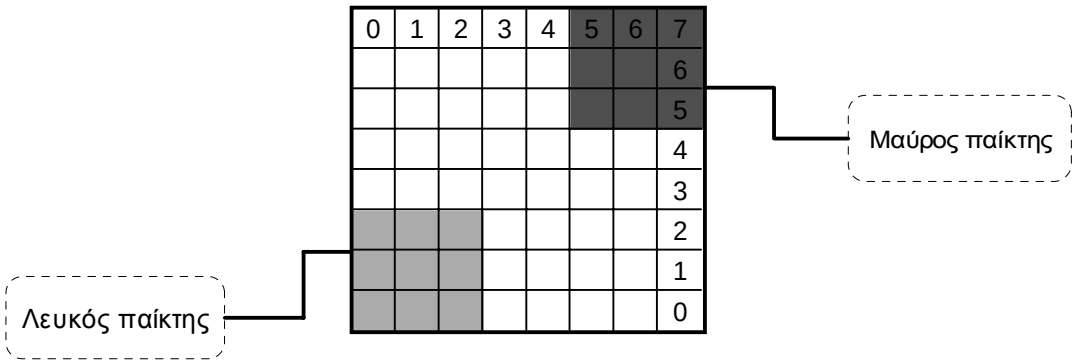
Οι παίκτες

Το παιχνίδι παίζεται από δύο παίκτες-αντιπάλους. Ο αντίπαλος μπορεί να είναι είτε κάποιος άνθρωπος, είτε ο υπολογιστής, συνεπώς υπάρχουν 3 διαφορετικά είδη παιχνιδιού:

- Υπολογιστής – Υπολογιστής
- Άνθρωπος – Υπολογιστής
- Άνθρωπος – Άνθρωπος

Ανεξάρτητα από το είδος του παιχνιδιού ονομάζουμε για λόγους τυποποίησης τους παίκτες άσπρος *(white)* και μαύρος *(black)*.

Η σκακιέρα πάνω στην οποία διεξάγεται το παιχνίδι φαίνεται στο ακόλουθο σχήμα (Σχήμα 1.1) :



Σχήμα 1.1 Η σκακιέρα του παιχνιδιού (διαστάσεις n=8, a=3)

Στα πλαίσια της διπλωματικής χρησιμοποίησαμε μία σκακιέρα διαστάσεων 8x2x10 (8: η διάσταση της σκακιέρας, 2: η διάσταση της βάσης, 10: το πλήθος των πιονιών του κάθε παίκτη)

1.3. Κανόνες κίνησης πιονιών

Μπορούμε να κατηγοριοποιήσουμε τις κινήσεις των πιονιών σε δύο κατηγορίες ως εξής:

- Οι κινήσεις εξόδου από τη βάση

Η κάθε βάση θεωρείται ως ένα τετράγωνο και όχι ως μια συλλογή τετραγώνων, συνεπώς οποιοδήποτε πιόνι της βάσης μπορεί να μετακινηθεί, με μια κίνηση, σε οποιοδήποτε από τα παρακείμενα στη βάση τετράγωνα που είναι ελεύθερα .

- Οι κινήσεις μετακίνησης από μία θέση σε μία άλλη

Κάθε πιόνι μπορεί να μετακινηθεί σε οποιοδήποτε (πάνω, κάτω, δεξιά, αριστερά) γειτονικό ελεύθερο τετραγωνάκι της σκακιέρας, με την προϋπόθεση ότι η μέγιστη διαφορά του από τη βάση του δεν μειώνεται (δεν επιτρέπονται δηλαδή κινήσεις προς τα πίσω).

Σε ένα σύστημα συντεταγμένων ο παραπάνω κανόνας θα μπορούσε να οριστεί ως εξής:

Αν *(x,y)* είναι η τρέχουσα θέση του πιονιού στην σκακιέρα, τότε αυτό θα μπορούσε να μετακινηθεί στη θέση

(x,z) αν και μόνο αν ισχύει:

$$\max(x - a, y - a) \leq \max(x - a, z - a)$$

στην περίπτωση που παίζει ο πρώτος παίκτης

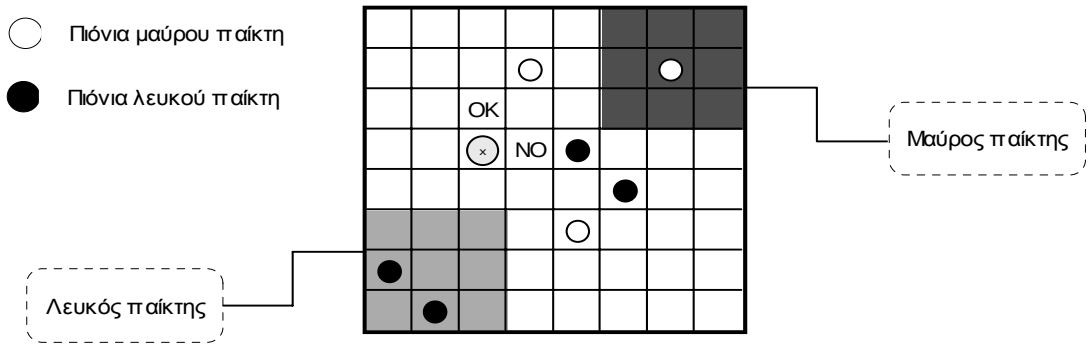
ή αν ισχύει:

$$\max(n - a - x, n - a - y) \leq \max(n - a - x, n - a - z)$$

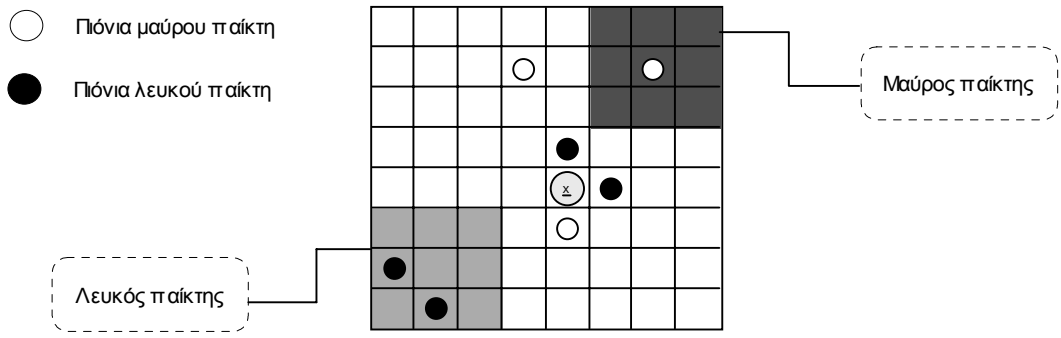
στην περίπτωση που παίζει ο δεύτερος παίκτης.

Εκτός από τις μη επιτρεπτές κινήσεις υπάρχουν και κινήσεις που προκαλούν την απώλεια των πιονιών. Ως τέτοιες ορίζουμε όλες εκείνες τις κινήσεις που επιφέρουν την άμεση γειτνίαση του κινούμενο πιονιού με κάποιο πιόνι του αντιπάλου. Σε τέτοιες περιπτώσεις το ‘’εγκλωβισμένο’’ πιόνι απομακρύνεται αυτόματα από τη σκακιέρα. Ομοίως, στην περίπτωση που δεν υπάρχει κανένα ελεύθερο παρακείμενο στη βάση τετραγωνάκι τα υπόλοιπα πιόνια της βάσης εξαφανίζονται αμέσως. Στο ακόλουθο σχήμα (Σχήμα 1.2)

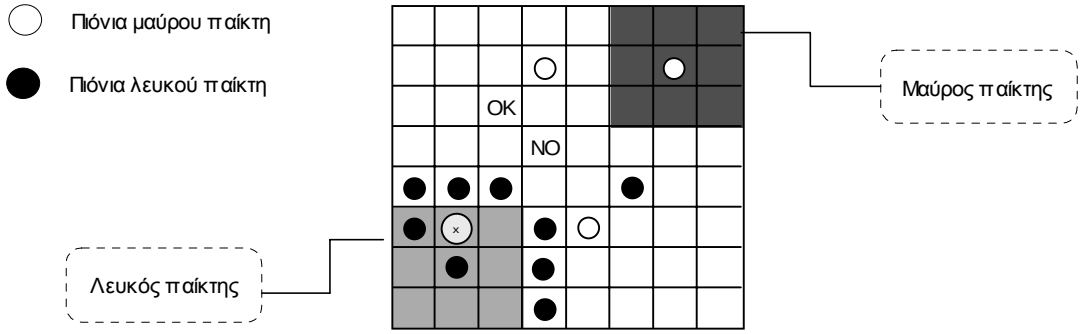
παρουσιάζουμε μερικά παραδείγματα μη επιτρεπτών κινήσεων και κινήσεων που προκαλούν την απώλεια κάποιου πιονιού.



(a) Παράδειγμα μη επιτρεπτής κίνησης, το πιόνι *x* που ανήκει στο μαύρο παίκτη δεν μπορεί να κινηθεί στη θέση ΝΟ, καθώς έτσι παραβιάζεται ο κανόνας της αυξανόμενης μέγιστης διαφοράς του πιονιού από τη βάση του.



(b) Παράδειγμα απώλειας πιονιού, δεν υπάρχει καμία επιτρεπτή κίνηση για το πιόνι *x* που εξαφανίζεται αυτόματα.

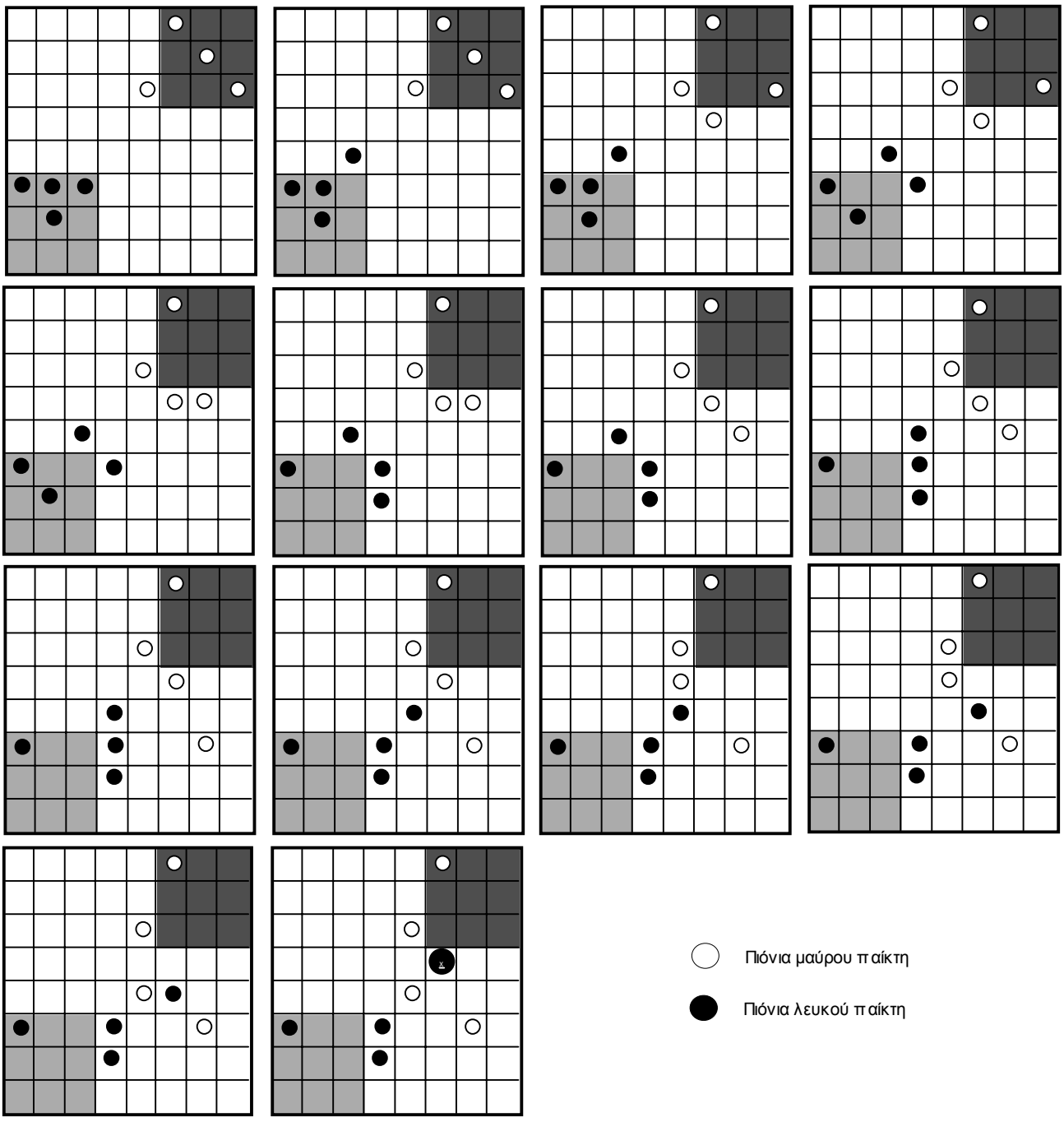


(c) Παράδειγμα απώλειας όλων των πιονιών της βάσης, τα εντός της βάσης πιόνια του λευκού παίκτη εξαφανίζονται αμέσως, καθώς δεν υπάρχει πλέον καμία επιτρεπτή κίνηση γι' αυτά.

Σχήμα 1.2 Παραδείγματα μη επιτρεπτών κινήσεων και κινήσεων που προκαλούν την απώλεια πιονιών.

1.4. Ένα παράδειγμα παιχνιδιού

Για καλύτερη κατανόηση των κανόνων του παιχνιδιού παραθέτουμε ένα πλήρες παράδειγμα παιχνιδιού. (Σχήμα 1.3)



Σχήμα 1.3 Παράδειγμα ενός πλήρους παιχνιδιού, ο λευκός παίκτης είναι ο νικητής καθώς με την επόμενη κίνησή του, ανεξάρτητα από την κίνηση που θα κάνει ο αντίπαλός του, έχει τη δυνατότητα να μπει στη βάση του μαύρου παίκτη με το πιόνι *x*.

2. Επιλογή στρατηγικής – Εισαγωγή στην Ενισχυτική Μάθηση

2.1. Θεωρία παιχνιδιών *(game theory)*

Η θεωρία παιχνιδιών αποτελεί εδώ και δεκαετίες έναν εξαιρετικά ενδιαφέροντα τομέα μελέτης και έρευνας. Οι επιστήμονες εστιάζουν σε παιχνίδια στρατηγικής και προσπαθούν με χρήση Τεχνητής Νοημοσύνης *(Artificial Intelligence)* να δημιουργήσουν ''έξυπνα'' προγράμματα που θα συναγωνίζονται ως προς την απόδοσή τους πραγματικούς παίκτες. Τέτοιου είδους παιχνίδια ενδείκνυνται για μελέτη λόγω της πολυπλοκότητάς τους και της έξυπνης στρατηγικής που χρειάζεται να ακολουθήσει κάποιος για να κερδίσει. Επιπλέον, οι είσοδοι του παιχνιδιού και τα κριτήρια αξιολόγησης είναι γνωστά, ενώ το περιβάλλον του παιχνιδιού, οι έγκυρες κινήσεις και οι κινήσεις τερματισμού μπορούν εύκολα να προσομοιωθούν.

Το 1995 ο Samuel [Samuel 1959] έφτιαξε ένα πρόγραμμα ντάμας *(checkers play)*, ενώ από το '60 ξεκίνησε η δημιουργία προγραμμάτων σκακιού. Στη δεκαετία του '90, η IBM, πρώτα με το Deep Though και στη συνέχεια με το Deep Blue, κατέβαλλε σφοδρές προσπάθειες για τη δημιουργία ενός προγράμματος σκακιού ικανού να ανταγωνίζεται τους καλύτερους σκακιστές του κόσμου. Ένα από τα πιο σημαντικά παιχνίδια στις μέρες μας είναι το TD-Gammon του Tesauro [Tesauro 1992], [Tesauro 1995] για το τάβλι.

Το σημαντικότερο και το πιο κρίσιμο σημείο ενός στρατηγικού παιχνιδιού είναι η επιλογή και η υλοποίηση της στρατηγικής που θα ακολουθήσει ο υπολογιστής κατά τη διάρκεια του παιχνιδιού. Με τον όρο στρατηγική εννοούμε την επιλογή της επόμενης κίνησής του υπολογιστή λαμβάνοντας υπόψη την παρούσα κατάστασή του, την κατάσταση του αντιπάλου, τις επιπτώσεις της κίνησής του και την ενδεχόμενη επόμενη κίνηση του αντιπάλου.

2.2. Κλασσικές προσεγγίσεις - Αλγόριθμος Min - Max

Μια κλασσική προσέγγιση στο παραπάνω πρόβλημα αποτελεί ο αλγόριθμος Min-Max [Samuel 1959] που στηρίζεται στη δημιουργία ενός δέντρου, οι κόμβοι του οποίου αντιστοιχούν στις καταστάσεις του παιχνιδιού ενώ οι ακμές του αντιστοιχούν στις ενδεχόμενες κινήσεις. Ψάχνοντας το δέντρο εις βάθος, η βέλτιστη κίνηση για τη δεδομένη κατάσταση υπολογίζεται. Αυτό μπορεί να γίνει βρίσκοντας τις πιθανές επόμενες κινήσεις μας, τις απαντήσεις του αντιπάλου, τις δικές μας απαντήσεις σε τέτοιο βάθος ώστε να βρούμε εκείνη την ακολουθία κινήσεων που θα μας οδηγήσει στη νίκη. Εμείς προσπαθούμε να αυξήσουμε το σκορ μας, γι' αυτό και καλούμαστε MAX, ενώ ο αντίπαλος προσπαθεί το αντίθετο, δηλαδή να μειώσει το σκορ του, γι' αυτό εξάλλου και καλείται MIN.

Ο ψευδοκώδικας του αλγορίθμου MIN-MAX είναι ο ακόλουθος:

Δημιούργησε όλο το δέντρο έρευνας για το παιχνίδι.

Η τιμή κάθε φύλλου καθορίζει την τιμή του πατέρα του *x*.

Αν είναι η σειρά του MAX τότε:

Τιμή του *x* ← η μέγιστη τιμή των παιδιών του.

Αν είναι η σειρά του MIN τότε:

Τιμή του *x* ← η μικρότερη τιμή των παιδιών του.

Επανέλαβε τις αναθέσεις τιμών στους κόμβους των παραπάνω επιπέδων μέχρι τη ρίζα του δέντρου

Ο MAX θα διαλέξει το παιδί της ρίζας με τη μέγιστη τιμή, ενώ ο MIN αυτό με την μικρότερη τιμή.

Ο MIN – MAX αλγόριθμος ψάχνει όλο το δέντρο του παιχνιδιού με αποτέλεσμα η υπολογιστική του πολυπλοκότητα να είναι πολύ μεγάλη. Ένα ακόμη μειονέκτημά του αλγορίθμου αποτελεί το ότι η τιμή στους κόμβους εξαρτάται από το βάθος του δέντρου οδηγώντας πολλές φορές σε λάθος εκτιμήσεις.

Μια παραλλαγή του MIN-MAX είναι η (a,β) μέθοδος κλαδέματος *((a-b) pruning method)* [Knuth & Moore 1975], μέσω της οποίας προσπαθούμε να μειώσουμε την πολυπλοκότητα του MIN-MAX. Προς την κατεύθυνση αυτή μειώνουμε το πλήθος των κόμβων που χρειάζεται να εξετάσει ο αλγόριθμος MIN-MAX με την ελπίδα ότι εξετάζοντας το κλαδεμένο δέντρο θα μπορέσουμε να υπολογίσουμε τη σωστή MIN-MAX απόφαση. Η λογική έχει ως εξής: Έστω *n* ένας κόμβος του δέντρου στον οποίο μπορούμε να μεταβούμε. Αν υπάρχει κάποια καλύτερη επιλογή στο γονέα ή σε κάποιο άλλο κόμβο πιο ψηλά τότε είναι προφανές ότι δε θα φτάσουμε ποτέ στον κόμβο *n* κατά τη διάρκεια του παιχνιδιού και συνεπώς μπορούμε να τον κλαδέψουμε.

Δεδομένων των προβλημάτων που παρουσιάζουν οι κλασσικοί αλγόριθμοι, χρησιμοποιούμε στην περίπτωση του παιχνιδιού μας μεθόδους Μηχανικής Μάθησης *(Machine Learning - ML)* και πιο συγκεκριμένα Ενισχυτική Μάθηση *(Reinforcement Learning - RL)* προκειμένου να βελτιώσουμε την απόδοση του υπολογιστή μέσω ενός πλήθους παιχνιδιών με τον εαυτό του. Αρκετά γνωστά παιχνίδια στρατηγικής χρησιμοποιούν Τεχνητή Νοημοσύνη με πιο γνωστό το TD-Gammon του Tesauro [Tesauro 1995], το οποίο αφού εκπαιδεύτηκε παίζοντας 1,5 εκατομμύρια παιχνίδια με τον εαυτό του έχει φτάσει πλέον στο επίπεδο να ανταγωνίζεται διεθνείς πρωταθλητές στο τάβλι. Αξίζει να αναφέρουμε επίσης και διάφορες παραλλαγές του: Tetris, Blackjack, Othello [Leouski 1995], Chess [Thrun 1995] που χρησιμοποιούν την ίδια προσέγγιση.

Ας δούμε όμως πρώτα τι είναι η Ενισχυτική Μάθηση *(Reinforcement Learning - RL).*

2.3. Ενισχυτική μάθηση *(Reinforcement learning - RL)*

Υπάρχουν πολλά προβλήματα, π.χ. συστήματα ελέγχου εναέριας κυκλοφορίας, που θα μπορούσαν να λύσουν οι υπολογιστές αν υπήρχε το κατάλληλο λογισμικό. Τα προβλήματα αυτά παραμένουν άλυτα όχι επειδή οι υπολογιστές στερούνται υπολογιστικής ισχύος αλλά επειδή είναι πολύ δύσκολο να καθορίσει κανείς τι πρέπει να κάνει το πρόγραμμα. Αν οι υπολογιστές μάθαιναν να επιλύουν τέτοια προβλήματα πειραματιζόμενοι μ' αυτά, το όφελος είναι προφανές. Η θεμελιώδης ιδέα της ενισχυτική μάθησης έχει την αφετηρία της στην ψυχολογία και πιο συγκεκριμένα στις πειραματικές μελέτες πάνω στη συμπεριφορά των ζώων. Ένας ορισμός της θα μπορούσε να είναι ο ακόλουθος:

> Αν ένα σύστημα μάθησης αναλαμβάνει μια ενέργεια (action) και στη συνέχεια ακολουθεί μια ικανοποιητική κατάσταση ενεργειών, τότε η τάση του συστήματος να παράγει αυτή τη συγκεκριμένη ενέργεια ενισχύεται. Διαφορετικά, η τάση του συστήματος να παράγει αυτή την ενέργεια αποδυναμώνεται.

Η ενισχυτική μάθηση *(RL)* συνδυάζει δύο πεδία, το Δυναμικό Προγραμματισμό *(Dynamic Programming)* και την Επιβλεπόμενη Μάθηση *(Supervised Learning)*.

Ο Δυναμικός Προγραμματισμός *(Dynamic Programming)* είναι πεδίο των μαθηματικών που παραδοσιακά χρησιμοποιείται σε προβλήματα βελτιστοποίησης και ελέγχου. Ωστόσο, υπάρχουν περιορισμοί ως προς το πλήθος και την πολυπλοκότητα των προβλημάτων που μπορεί να αντιμετωπίσει.

Η Επιβλεπόμενη Μάθηση *(Supervised Learning)* είναι μια γενική μέθοδος εκπαίδευσης μιας παραμετροποιημένης προσέγγισης μιας συνάρτησης. Ωστόσο για να μάθει κανείς τη συνάρτηση, χρειάζεται να γνωρίζει σωστά παραδείγματα εισόδου - εξόδου που στην πράξη είναι δύσκολο να υπάρχουν πάντα.

Για τους λόγους αυτούς, αναπτύχθηκε η Ενισχυτική Μάθηση *(Reinforcement Learning -RL)* που αποτελεί μια ορθογώνια σχεδόν προσέγγιση στις άλλες μεθόδους μηχανικής μάθησης, αν και όπως θα δούμε στη συνέχεια μπορεί να συνυπάρξει τουλάχιστον με τα νευρωνικά δίκτυα.

Η Ενισχυτική Μάθηση είναι συνώνυμη της αλληλεπιδραστικής μάθησης. Ο υπολογιστής έχει απλώς να επιτύχει κάποιο στόχο - μόνος του μαθαίνει πως θα καταφέρει κάτι τέτοιο πειραματιζόμενος και κάνοντας λάθη κατά την αλληλεπίδραση του με το περιβάλλον *(trial and error learning)*.

Τα πλεονεκτήματα της Ενισχυτικής Μάθησης σε σχέση με τις παραδοσιακές προσεγγίσεις στη μάθηση είναι πολλά. Καταρχήν, η Ενισχυτική Μάθηση απαιτεί πολύ λίγη προγραμματιστική προσπάθεια καθώς η εκπαίδευση του συστήματος είναι αυτόματη και προκύπτει από την αλληλεπίδραση του με το περιβάλλον. Επιπλέον, το σύστημα αντιλαμβάνεται τυχόν αλλαγές στο περιβάλλον μάθησης χωρίς να χρειάζεται να προγραμματιστεί εκ νέου, η ιδιότητα αυτή αποδίδεται στη γενικότητα της Ενισχυτικής Μάθησης.

Η Ενισχυτική Μάθηση μοιάζει με τους Γενετικούς Αλγορίθμους *(Genetic Algorithms)* στο γεγονός ότι στηρίζονται και οι δύο στην εμπειρία – εκπαίδευση και όχι στον προγραμματισμό με αποτέλεσμα να προσαρμόζονται εύκολα στις τυχόν αλλαγές του περιβάλλοντος. Το πλεονέκτημα της Ενισχυτικής Μάθησης έναντι των Γενετικών έγκειται στο ότι η Ενισχυτική Μάθηση χρησιμοποιεί καλύτερα τους πόρους που διαθέτει. Για παράδειγμα, αν μια στρατηγική ήταν πολύ φτωχή ο γενετικός αλγόριθμος θα την απέρριπτε και θα κρατούσε για τις επόμενες γενιές στρατηγικές με μεγαλύτερη αξία. Στην περίπτωση της Ενισχυτικής Μάθησης όμως, αν δοθείσας μιας κατάστασης *s* για την εν λόγω στρατηγική αναλαμβανόταν μια ενέργεια *a* και η νέα κατάσταση που ακολουθούσε έχει πάντα μεγάλη αξία τότε το σύστημα θα είχε μάθει πως παρόλο που η ίδια η στρατηγική δεν είναι καλή η ενέργεια *a* από την κατάσταση *s* είναι καλή.

2.4. Trial and error learning (Παράδειγμα εκμάθησης ποδηλάτου)

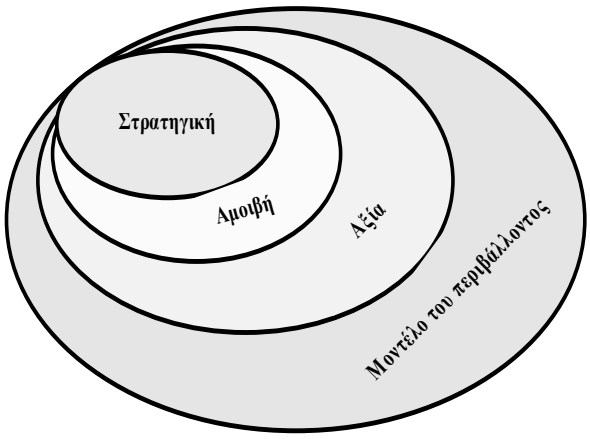
Ας μελετήσουμε αδρά τις αρχές της Ενισχυτικής Μάθησης *(Reinforcement Learning - RL)* στο πρόβλημα της εκμάθησης ποδηλάτου για να καταλάβουμε καλύτερα τη φιλοσοφία της. Στόχος του συστήματος ενισχυτικής μάθησης είναι να μάθει να οδηγεί το ποδήλατο χωρίς να πέσει κάτω. Στην πρώτη προσπάθεια, το σύστημα κάνει κάποιες κινήσεις που προκαλούν κλίση 45 μοιρών δεξιά στο ποδήλατο. Στην παρούσα κατάσταση μπορεί ή να στρίψει αριστερά ή να στρίψει δεξιά. Επιλέγει το πρώτο και αμέσως πέφτει, λαμβάνοντας έτσι ισχυρά αρνητική αμοιβή *(reward)*. Το σύστημα έχει πλέον μάθει να μην στρίβει αριστερά όταν βρίσκεται 45 μοίρες δεξιά. Στην επόμενη προσπάθεια, μετά από αρκετές κινήσεις το σύστημα καταλήγει και πάλι να βρίσκεται 45 μοίρες δεξιά. Γνωρίζει από την πρώτη προσπάθεια ότι δεν πρέπει να στρίψει αριστερά και επιλέγει τη μόνη δυνατή κίνηση, στρίβει δεξιά. Και πάλι, πέφτει λαμβάνοντας ισχυρά αρνητική ενίσχυση. Στο σημείο αυτό το σύστημα δεν έχει μάθει απλώς ότι όταν βρίσκεται 45 μοίρες δεξιά δεν είναι καλό να κινηθεί ούτε δεξιά ούτε αριστερά, αλλά επιπλέον έμαθε πως το να βρίσκεται σε κατάσταση 45 μοιρών δεξιά δεν είναι καλό. Στην επόμενη προσπάθεια, εκτελεί κάποιες κινήσεις και καταλήγει να βρίσκεται 40 μοίρες δεξιά. Στρίβει αριστερά με αποτέλεσμα να βρεθεί σε κατάσταση 45 μοιρών δεξιά και λαμβάνει ισχυρά αρνητική ενίσχυση. Το σύστημα μόλις έμαθε να μην στρίβει αριστερά όταν βρίσκεται 40 μοίρες δεξιά. Κάνοντας πολλές τέτοιες προσπάθειες το σύστημα θα μάθει πως να αποτρέπει το ποδήλατο από την πτώση.

2.5. Στοιχεία της Ενισχυτικής Μάθησης (RL)

Πέντε είναι τα βασικά στοιχεία ενός RL συστήματος:

1. **Ο μαθητής – πράκτορας *(Agent)*:** Είναι αυτός που μαθαίνει μέσω του RL συστήματος. Μπορεί να εκτελεί κάποιες κινήσεις $a \in A$, όπου Α το σύνολο των επιτρεπτών κινήσεων, ανάλογα με την κατάσταση του περιβάλλοντος στην οποία βρίσκεται.
2. **Το μοντέλο του περιβάλλοντος *(Model of the environment)*:** Μιμείται τη συμπεριφορά του περιβάλλοντος, δηλ. δοθείσας μιας κατάστασης και μιας κίνησης καθορίζει ποια θα είναι η επόμενη κίνηση. Το περιβάλλον περιγράφεται από ένα σύνολο καταστάσεων $s \in S$, όπου *S* το σύνολο των καταστάσεων του περιβάλλοντος.
3. **Η στρατηγική *(Policy) π*:** Καθορίζει τη συμπεριφορά του μαθητή *(agent)* σε κάθε χρονική στιγμή, δηλαδή τι πρέπει να κάνει. Πρόκειται για μία αντιστοιχία διακριτών καταστάσεων του περιβάλλοντος σε κινήσεις που θα πρέπει να ληφθούν από τον agent όταν περιέλθει σε μια τέτοια κατάσταση.
4. **Η συνάρτηση αμοιβής *(Reward function):*** Αντιστοιχεί κάθε κατάσταση του συστήματος (ή ζεύγη καταστάσεων - κινήσεων) σε έναν αριθμό, την αμοιβή (*reward) r*, που εκφράζει αν η συγκεκριμένη κατάσταση είναι επιθυμητή. Η συνάρτηση αμοιβής καθορίζει κατά κάποιο τρόπο ποιες από τις καταστάσεις στις οποίες ενδέχεται να βρεθεί ο agent είναι καλές.
5. **Η συνάρτηση αποτίμησης *(Value function):*** Καθορίζει ποιες κινήσεις είναι καλές μακροπρόθεσμα. Πιο συγκεκριμένα, η τιμή μιας κατάστασης υπολογίζεται ως το άθροισμα των αμοιβών που αναμένεται να συγκεντρώσει ο agent μέχρι το τέλος του παιχνιδιού. Σε αντίθεση με τις αμοιβές που εκφράζουν την προσωρινή αξία μιας κατάστασης του περιβάλλοντος, οι τιμές εκφράζουν την μακροπρόθεσμη αξία μιας κατάστασης λαμβάνοντας υπόψη τις καταστάσεις που ενδέχεται να προκύψουν στη συνέχεια και τις αντίστοιχες αμοιβές. Οι αμοιβές δίνονται κατευθείαν από το περιβάλλον, ενώ οι τιμές υπολογίζονται συνέχεια και ενημερώνονται βάσει των παρατηρήσεων του agent κατά την αλληλεπίδρασή του με το περιβάλλον.

Τα πιο σημαντικά στοιχεία του RL συνοψίζονται με γραφικό τρόπο στο ακόλουθο σχήμα:

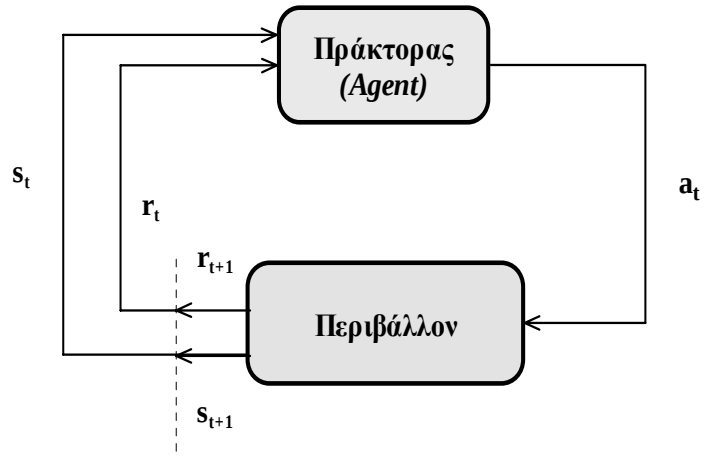


Σχήμα 2.1 Τα πιο σημαντικά στοιχεία ενός RL πράκτορα

2.6. Περιβάλλον μάθησης

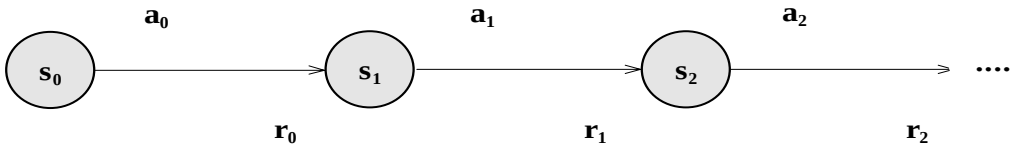
Ο agent αλληλεπιδρά με το περιβάλλον μάθησης κάθε χρονική στιγμή $t = 0,1,2\ldots$ (θεωρούμε διακριτό χρόνο για λόγους απλότητας και κατανόησης – θα μπορούσαν οι ίδιες ιδέες να επεκταθούν και στην

περίπτωση του συνεχούς χρόνου). Πιο συγκεκριμένα, τη χρονική στιγμή *t* ο agent βλέποντας ότι το περιβάλλον βρίσκεται στην κατάσταση $s \in S$ εκτελεί την κίνηση $a \in A_t$,όπου A_t είναι το σύνολο των επιτρεπτών κινήσεων τη χρονική στιγμή *t* για τη δεδομένη κατάσταση s του περιβάλλοντος. Την αμέσως επόμενη στιγμή *t+1* ο agent λαμβάνει άμεση αμοιβή *(reward)* $r_{t+1} \in \Re$ ως αποτέλεσμα της κίνησης που έκανε μόλις πριν και μεταβαίνει σε μια νέα κατάσταση s_{t+1}. Στο επόμενο σχήμα (Σχήμα 2.2) φαίνεται καθαρά η αλληλεπίδραση πράκτορος - περιβάλλοντος.



Σχήμα 2.2 Αλληλεπίδραση πράκτορος *(agent)* - περιβάλλοντος

Η αλληλεπίδραση αυτή δημιουργεί ένα σύνολο καταστάσεων $s_{i,}$ κινήσεων a_i και άμεσων αμοιβών *(rewards)* r_i (Σχήμα 2.3). Στόχος του πράκτορα *(agent)* είναι να μάθει μια στρατηγική ελέγχου $\pi : S \rightarrow A$, η οποία να μεγιστοποιεί το αναμενόμενο άθροισμα των αμοιβών $r_{i,}$ λαμβάνοντας υπόψη και τη χρονική στιγμή *i* στην οποία αποδόθηκε η αμοιβή.



Σχήμα 2.3 Ακολουθία καταστάσεων – κινήσεων και άμεσων αμοιβών *(rewards)* στο RL

Πιο φορμαλιστικά, ο agent επιλέγει κινήσεις a_t που μεγιστοποιούν την ποσότητα:

$$R_t = r_{t+1} + \gamma * r_{t+2} + \gamma^2 * r_{t+3} + \ldots = \sum \gamma^k * r_{t+k+1}$$

όπου:

R_t : η αναμενόμενη αμοιβή τη χρονική στιγμή t

γ: η παράμετρος του ρυθμού μείωσης *(discount rate parameter)*, μια σταθερά με τιμές *0≤γ<1.*

Η παρουσία του *γ* καθορίζει την αξία μελλοντικών αμοιβών. Μία αμοιβή που λαμβάνεται τη χρονική στιγμή *k* έχει αξία μικρότερη κατά γ^{k-1} σε σχέση με την αξία που θα είχε αν λαμβανόταν την τρέχουσα χρονική στιγμή *t*.

Αν *γ<1,* η ακολουθία R_t συγκλίνει μόνο αν η ακολουθία των αμοιβών $\{r_t\}$ είναι φραγμένη.

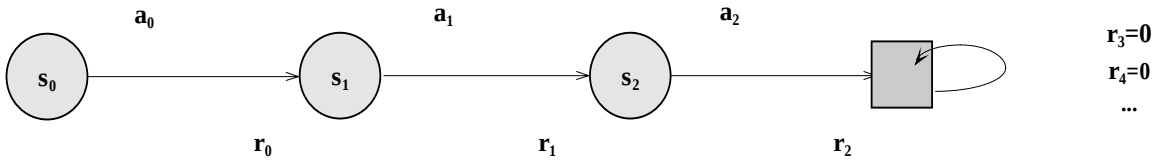
Αν *γ=0,* ο agent ενδιαφέρεται για την άμεση μεγιστοποίηση των αμοιβών, είναι κοντόφθαλμος-βραχυπρόθεσμη στρατηγική.

Αν *γ→1,* ο agent λαμβάνει σοβαρά υπόψη του τις μελλοντικές αμοιβές, είναι πιο δίκαιος - μακροπρόθεσμη στρατηγική.

Η επιλογή της κίνησης, δεδομένης της κατάστασης του περιβάλλοντος, είναι πολύ σημαντική και επηρεάζει καθοριστικά τη διαδικασία μάθησης. Ο agent αντιμετωπίζει το εξής δίλημμα: να επιλέξει κάποια κίνηση για την οποία έχει ήδη μάθει ότι έχει υψηλή αμοιβή ή να δοκιμάσει καινούριες κινήσεις για τις οποίες δεν γνωρίζει τίποτα αλλά μπορεί να αποδειχτούν πιο επιθυμητές (με μεγαλύτερη δηλαδή αμοιβή). Στην πρώτη περίπτωση ο πράκτορας *(agent)* εκμεταλλεύεται στην ουσία τη γνώση που ήδη κατέχει και επιλέγει κινήσεις που μεγιστοποιούν άμεσα την αμοιβή *(*η περίπτωση αυτή αναφέρεται ως *exploitation).* Ενώ στη δεύτερη περίπτωση, ο agent επιλέγει να ρισκάρει δοκιμάζοντας νέες κινήσεις που μπορεί να έχουν μεγαλύτερη αμοιβή αλλά μπορεί και όχι *(*η περίπτωση αυτή αναφέρεται ως *exploration).*

Η απάντηση είναι ότι πρέπει να τα συνδυάσει και τα δύο, αν θέλει να μεγιστοποιήσει τη συνολική αμοιβή. Μια συνηθισμένη τακτική είναι στην αρχή ο agent να ανακαλύπτει τις πλέον συμφέρουσες κινήσεις *(exploration)* και στη συνέχεια να εκμεταλλεύεται την αποκτηθείσα γνώση επιλέγοντας τις πιο συμφέρουσες κινήσεις *(exploitation).*

Η διαδικασία της μάθησης σταματά όταν ο agent εισέλθει σε μία ειδική κατάσταση, γνωστή ως κατάσταση απορρόφησης ή κατάσταση τερματισμού *(absorbing state)*, στην οποία οι μόνες δυνατές κινήσεις οδηγούν και πάλι στην ίδια κατάσταση και η αμοιβή κάθε κίνησης είναι μηδέν (Σχήμα 2.4).



Σχήμα 2.4 Κατάσταση τερματισμού *(absorbing state)*

Μερικές φορές η αμοιβή δίνεται καθυστερημένα στον agent (π. χ στο τέλος μιας ακολουθίας εισόδων - εξόδων). Στις περιπτώσεις αυτές ο μαθητής καλείται να αντιμετωπίσει το γνωστό ως «ανάθεση προσωρινής πίστωσης» πρόβλημα *(temporal credit assignment problem),* βλέπε Κεφάλαιο 5 για παραπάνω λεπτομέρειες. Καλείται δηλαδή να εκτιμήσει κατά πόσο οι διάφορες είσοδοι – έξοδοι ευθύνονται, είτε αρνητικά είτε θετικά, για την τελική αμοιβή. Το πρόβλημα της ανάθεσης προσωρινής πίστωσης εξακολουθεί να θεωρείται από τα πλέον δύσκολα προβλήματα της ενισχυτικής μάθησης.

2.7. Η ιδιότητα Markov

Οι καταστάσεις του περιβάλλοντος έτσι όπως παρουσιάζονται στον agent δεν αποτελούν πλήρεις περιγραφές του περιβάλλοντος. Υπάρχει κρυμμένη πληροφορία, η οποία δεν παρουσιάζεται μέσω των καταστάσεων. Η γνώση του agent για το περιβάλλον είναι ελλιπής, και γι' αυτό δεν ευθύνεται ο agent. Ο agent είναι υπεύθυνος να θυμάται πράγματα που έχει μάθει αλλά σε καμία περίπτωση δεν μπορούμε να τον κατηγορήσουμε για πράγματα που δεν θα μπορούσε να μάθει με τα διαθέσιμα δεδομένα.

Το ιδανικό θα ήταν η τρέχουσα κατάσταση του περιβάλλοντος να αντανακλούσε και τις προγενέστερές της καταστάσεις. Μία τέτοια κατάσταση λέμε ότι έχει την ιδιότητα Markov. Πιο φορμαλιστικά, μια κατάσταση *s* έχει την ιδιότητα Markov αν ισχύει :

$$\Pr\{s_{t+1}=s', r_{t+1}=r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, \ldots r_1, s_0, a_0\} = \Pr\{s_{t+1}=s', r_{t+1}=r \mid s_t, a_t\}$$

Αν η παραπάνω σχέση ισχύει για όλα τα *s',r* και τις προηγούμενες κινήσεις $s_t,a_t,r_t \ldots r_1,s_0,a_0$ λέμε ότι και το περιβάλλον έχει την ιδιότητα Markov. Συνεπώς δοθέντων της τρέχουσας κατάστασης και της λαμβανόμενης κίνησης μπορούμε να προβλέψουμε την επόμενη κατάσταση και την αναμενόμενη αμοιβή.

Η ιδιότητα Markov είναι πολύ σημαντική στο RL, καθώς οι κινήσεις και οι αμοιβές που αποδίδονται εξαρτώνται μόνο από την τρέχουσα κατάσταση. Ένα RL πρόβλημα που ικανοποιεί την ιδιότητα Markov ονομάζεται **Μαρκοβιανή Διαδικασία Αποφάσεων** (*Markov Decision Process, MDP*). Στην περίπτωση που το σύνολο των καταστάσεων S και το σύνολο των κινήσεων A είναι πεπερασμένο, ονομάζεται **Πεπερασμένη Μαρκοβιανή Διαδικασία Αποφάσεων** (Finite MDP). Το παιχνίδι μας αποτελεί μια Πεπερασμένη Μαρκοβιανή Διαδικασία Αποφάσεων. Δοθέντων μιας κατάστασης *s* και μιας κίνησης *a*, η πιθανότητα μιας κατάστασης *s'* είναι:

$$P^{a}_{ss'} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$$

και η αναμενόμενη αμοιβή είναι:

$$R^{a}_{ss'} = E\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s\}$$

2.8. Συναρτήσεις αξιολόγησης (value functions)

Οι περισσότεροι αλγόριθμοι ενισχυτικής μάθησης βασίζονται στην εκτίμηση συναρτήσεων αξιολόγησης που εκτιμούν πόσο καλό είναι για τον agent να βρίσκεται σε μια συγκεκριμένη κατάσταση. Το ''πόσο καλό'' καθορίζεται με βάση τις μελλοντικές αμοιβές που αναμένεται να λάβει ο agent. Επειδή οι αμοιβές εξαρτώνται από τις κινήσεις, οι συναρτήσεις αξιολόγησης ορίζονται ως προς συγκεκριμένες στρατηγικές *π*.

Η αξία της κατάστασης *s* μιας στρατηγικής *π,* συμβολίζεται με $V^{\pi}(s)$, ισούται με το άθροισμα των αναμενόμενων μελλοντικών αμοιβών ξεκινώντας από την κατάσταση *s* και ακολουθώντας στη συνέχεια τη στρατηγική π. Πιο φορμαλιστικά:

$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k * r^{t+k+1} \mid s_t = s\}$$

$V^{\pi}(s)$ *:* η συνάρτηση αξιολόγησης καταστάσεων *(state-value function)* της στρατηγικής *π*.

E_{π}: η αναμενόμενη τιμή δεδομένου ότι ο agent ακολουθεί την πολιτική *π.*

Με τον ίδιο τρόπο, ορίζουμε την αξία μιας κίνησης *a* δεδομένης της τρέχουσας κατάστασης *s* και τη συμβολίζουμε με $Q^{\pi}(s,a)$. Το $Q^{\pi}(s,a)$ αναφέρεται στην στρατηγική *(policy) π* και ισούται με το άθροισμα των αναμενόμενων μελλοντικών αμοιβών ξεκινώντας από την κατάσταση *s*, επιτελώντας την κίνηση *a* και ακολουθώντας στη συνέχεια τη στρατηγική *π*.

Πιο φορμαλιστικά:

$$Q^{\pi}(s,a) = E_{\pi}\{R_t \mid s_t = s, a_t = a\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k * r^{t+k+1} \mid s_t = s, \alpha_t = \alpha\}$$

$Q^{\pi}(s,a)$ *:* η συνάρτηση αξιολόγησης κινήσεων *(action-value function)* της στρατηγικής *π*.

Οι συναρτήσεις $V^{\pi}(s)$, $Q^{\pi}(s,a)$ μπορούν να υπολογιστούν από την εμπειρία. Για παράδειγμα, ο agent θα μπορούσε ακολουθώντας την πολιτική *π* να κρατάει ένα μέσο όρο με την αναμενόμενη αμοιβή για κάθε κατάσταση που συναντά. Καθώς το πλήθος των επαναλήψεων θα έτεινε στο άπειρο ο μέσος όρος θα συνέκλινε στην πραγματική τιμή $V^{\pi}(s)$. Η ίδια διαδικασία θα μπορούσε να ακολουθηθεί και στην περίπτωση του $Q^{\pi}(s,a)$.

Οι μέθοδοι αυτές ονομάζονται Monte-Carlo μέθοδοι, επειδή ακριβώς εμπεριέχουν μέσους όρους τυχαίων παραδειγμάτων. Ένα μειονέκτημα τους είναι ότι στην περίπτωση πολλών καταστάσεων η πολυπλοκότητά τους αυξάνεται σημαντικά.

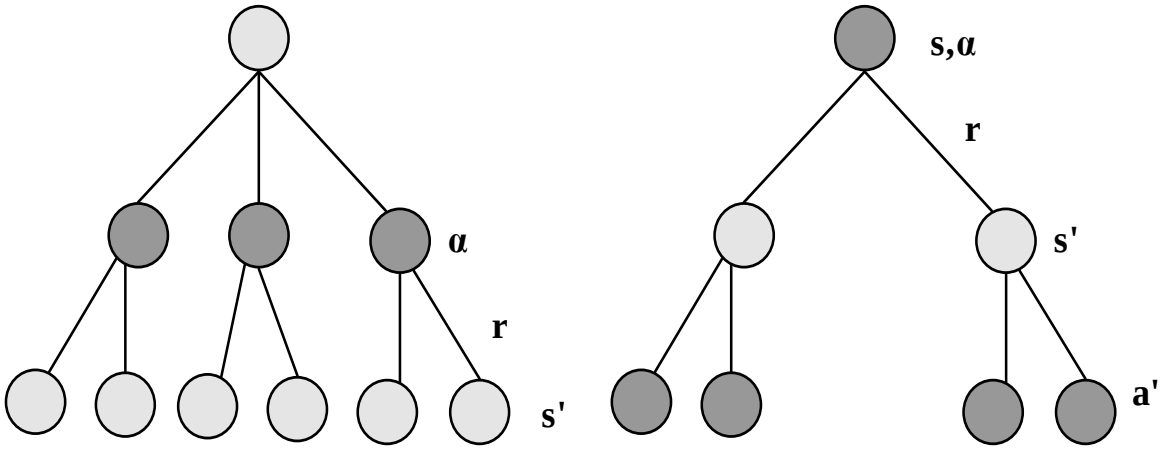
Ένα πολύ σημαντικό χαρακτηριστικό των $V^{\pi}(s)$, $Q^{\pi}(s,a)$ είναι η αναδρομική τους ιδιότητα (Εξίσωση Bellman) που εκφράζει μία σχέση μεταξύ μιας κατάστασης και των πιθανών αμέσως επόμενων καταστάσεων και ορίζεται ως εξής:

$$V^{\pi}(s) = \sum_{a} \pi(s, a) \sum_{s'} P^{a}_{ss'} [R^{a}_{ss'} + \gamma * V^{\pi}(s')]$$

s': κάποια από τις πιθανές επόμενες καταστάσεις της *s*.

Ας δούμε όμως την εφαρμογή της εξίσωσης Bellman στην πράξη (Σχήμα 2.5). Έστω μία κατάσταση *s* με τρεις πιθανές αμέσως επόμενες καταστάσεις. Ξεκινώντας από την *s* ο agent μπορεί να επιλέξει να μεταβεί σε μία από τις τρεις καταστάσεις. Έστω ότι επιλέγει την *s'* και λαμβάνει από το περιβάλλον αμοιβή *r* για την κίνηση του.

Τα διαγράμματα του σχήματος καλούνται backup διαγράμματα, ακριβώς επειδή αφορούν την ενημέρωση *(update).* Η ενημέρωση προκύπτει από τη μεταφορά πληροφορίας από τις προκύπτουσες καταστάσεις στην αρχική κατάσταση *s.*



Σχήμα 2.5 α) Backup διάγραμμα για το $V^{\pi}(s)$ **β)** Backup διάγραμμα για το $Q^{\pi}(s,a)$

Η εξίσωση Bellman υπολογίζει το μέσο όρο όλων των δυνατών καταστάσεων με βάση και την πιθανότητά τους να συμβούν. Θα πρέπει η αξία της αρχικής κατάστασης να ισούται με την (μειωμένη) αξία της αναμενόμενης επόμενης κατάστασης συν την αμοιβή που αναμένεται να λάβει ο πράκτορας *(agent)* για την κίνηση αυτή.

Η συνάρτηση αξιολόγησης $V^{\pi}(s)$ αποτελεί τη μοναδική λύση της εξίσωσης Bellman, δυστυχώς όμως η πολυπλοκότητά της είναι μεγάλη. Για το λόγο αυτό χρησιμοποιούνται οι βέλτιστες συναρτήσεις αξιολόγησης.

2.9. Βέλτιστες Συναρτήσεις αξιολόγησης (optimal value functions)

Για να επιλύσουμε ένα πρόβλημα reinforcement learning αρκεί να βρούμε μία τακτική που μακροπρόθεσμα θα μας εξασφαλίσει συνολικά μεγάλη αμοιβή. Στην περίπτωση των πεπερασμένων μαρκοβιανών διαδικασιών αποφάσεων *(Finite MDP)* μπορούμε να ορίσουμε μια βέλτιστη τακτική ως εξής:

Μία τακτική *π* καλείται βέλτιστη τακτική (optimal policy *π**) αν ισχύει:

$$V^{\pi}(s) \geq V^{\pi'}(s) \text{ για όλα τα } s \in S.$$

Οι συναρτήσεις $V^{\pi}(s)$, $Q^{\pi}(s,a)$ στην περίπτωση της βέλτιστης τακτικής *π** καλούνται βέλτιστη συνάρτηση αξιολόγησης καταστάσεων *(optimal state-value function)* και βέλτιστη συνάρτηση αξιολόγησης κινήσεων *(optimal action-value function)* αντίστοιχα. Συμβολίζονται με $V^{*}(s)$, $Q^{*}(s,a)$ και ορίζονται ως εξής:

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) \text{ για όλα τα } s \in S.$$

$$Q^{*}(s,\alpha) = \max_{\pi} Q^{\pi}(s) \text{ για όλα τα } s \in S \text{ , } a \in A(s).$$

Η εξίσωση Bellman στην περίπτωση της βέλτιστης τακτικής τροποποιείται ως εξής:

$$V^{*}(s) = \max_{\alpha} \sum_{s'} P^{\alpha}_{ss'} [R^{A}_{SS'} + \gamma V^{*}(s')] \text{ (Εξίσωση βελτιστοποίησης Bellman)}$$

Αν πρόκειται για πεπερασμένες MDP, η εξίσωση βελτιστοποίησης Bellman έχει μοναδική λύση. Πρόκειται για ένα σύστημα μη γραμμικών εξισώσεων όπου κάθε εξίσωση αντιστοιχεί σε μια κατάσταση. Αν π.χ. υπάρχουν N καταστάσεις θα δημιουργηθεί ένα σύστημα N εξισώσεων με N αγνώστους. Επιλύοντας το σύστημα των εξισώσεων μπορεί κανείς να βρει τη βέλτιστη τακτική. Μία τέτοια επίλυση όμως, περιλαμβάνει εξοντωτικό ψάξιμο όλων των πιθανών τακτικών και υπολογισμό των πιθανοτήτων εμφάνισής τους και τις αξίας τους όσον αφορά την αμοιβή που αναθέτουν στον agent. Μια τέτοια λύση εξαρτάται από τρεις τουλάχιστον παράγοντες:

- Ακριβή γνώση του περιβάλλοντος
- Επαρκείς υπολογιστικούς πόρους
- Ιδιότητα Markov

που είναι δύσκολο να πληρούνται στην πράξη. Για το λόγο αυτό καταφεύγουμε σε προσεγγιστικές λύσεις.

2.10. Βελτιστοποίηση και διαδικασία προσέγγισης

Μόλις προαναφέραμε πως η πολυπλοκότητα του υπολογισμού της βέλτιστης τακτικής είναι απαγορευτική καθώς η μνήμη που απαιτείται για την προσέγγιση των συναρτήσεων αξιολόγησης, των τακτικών και των μοντέλων του περιβάλλοντος είναι τεράστια. Βέβαια, στην περίπτωση προβλημάτων με μικρό, πεπερασμένο σύνολο καταστάσεων μπορούμε να υπολογίσουμε αυτές τις προσεγγίσεις χρησιμοποιώντας πίνακες με μία είσοδο για κάθε κατάσταση. Η περίπτωση αυτή καλείται περίπτωση υπολογιζόμενη σε πίνακα *(tabular case)* και αναφέρεται σε προβλήματα με μικρό πλήθος καταστάσεων.

Στην πράξη ωστόσο παρουσιάζουν ενδιαφέρον προβλήματα με μεγάλο σύνολο καταστάσεων για την επίλυση των οποίων θα πρέπει να καταφεύγουμε σε προσεγγιστικές λύσεις. Για παράδειγμα, κατά την διαδικασία προσέγγιση της βέλτιστης τακτικής, μπορεί να υπάρχουν πολλές καταστάσεις στις οποίες

σπάνια βρίσκεται ο agent, με αποτέλεσμα η επιλογή μη βέλτιστων κινήσεων να επηρεάζει ελάχιστα τη συνολική αμοιβή που λαμβάνει ο agent. Χαρακτηριστικό παράδειγμα αποτελεί το TD-Gammon του Tesauro που παρόλο που μπορεί να συναγωνίζεται παγκόσμιους πρωταθλητές, αποτυγχάνει σε περιπτώσεις καταστάσεων παιχνιδιού που δεν εμφανίζονται ποτέ στα παιχνίδια με τους πολύ καλούς παίκτες.

Η φύση του RL επιτρέπει την προσέγγιση εκείνων των βέλτιστων τακτικών που εστιάζουν στη λήψη καλών αποφάσεων για τις πιο συχνά εμφανιζόμενες καταστάσεις και δίνουν λιγότερη σημασία στις καταστάσεις που εμφανίζονται πιο σπάνια. Μερικά παραδείγματα συναρτήσεων προσέγγισης αποτελούν: τα perceptrons πολλών επιπέδων *(multi-layer perceptrons)*, τα στηριζόμενα στη μνήμη συστήματα *(memory based systems)*, οι Radial basis functions, οι πίνακες ψαξίματος στοιχείων *(lookup tables)* και άλλα.

3. Μάθηση Χρονικών Διαφορών*(Temporal difference learning)*

3.1. Εισαγωγή

Η μάθηση χρονικών διαφορών *(TD learning)* [Sutton & Barto 1998], [Sutton 1988], [Teasuro 1995] θεωρείται ως η πιο θεμελιώδης και ενδιαφέρουσα προσέγγιση στην ενισχυτική μάθηση. Οι TD μέθοδοι είναι γενικοί αλγόριθμοι μάθησης που χρησιμοποιούνται για μακροπρόθεσμες προβλέψεις *(prediction)* σε δυναμικά συστήματα. Οι ρυθμίσεις των παραμέτρων του συστήματος πρόβλεψης *(predictor)* γίνονται βάσει της διαφοράς (ή λάθους) μεταξύ χρονικών διαδοχικών επιτυχημένων εκτιμήσεων *(temporal successive predictions).* Έτσι, η μάθηση στις TD μεθόδους παρουσιάζεται όταν υπάρχει over time αλλαγή στην πρόβλεψη και αποσκοπεί στην μείωση της διαφοράς μεταξύ της εκτίμησης του πράκτορα για την τρέχουσα είσοδο και της εκτίμησης του πράκτορα για την αμέσως επόμενη χρονική στιγμή.

Η μάθηση χρονικών διαφορών συνδυάζει ιδέες Monte Carlo με ιδέες Δυναμικού Προγραμματισμού *(Dynamic Programming).* Όπως οι μέθοδοι Monte Carlo, οι TD μέθοδοι μαθαίνουν κατευθείαν από την εμπειρία χωρίς να χρειάζεται να γνωρίζουν το μοντέλο του περιβάλλοντος. Έχοντας κάποια εμπειρία σχετικά με την τακτική π, και οι δύο μέθοδοι ανανεώνουν τον υπολογισμό τους *V* για την V^{π} . Αν η κατάσταση που επισκέπτονται τη χρονική στιγμή t δεν είναι τελική, τότε και οι δύο μέθοδοι ανανεώνουν τον υπολογισμό τους για την $V(s_t)$ βασιζόμενοι στις συνέπειες αυτής της επίσκεψης.

Σε αντίθεση όμως με τις μεθόδους Monte Carlo, οι TD μέθοδοι δεν χρειάζεται να περιμένουν μέχρι το τέλος του επεισοδίου για να καθορίσουν την αύξηση στην τιμή του $V(s_t)$, αρκεί να περιμένουν μέχρι την επόμενη χρονική στιγμή. Τη χρονική στιγμή *t+1* σχηματίζουν αυτόματα έναν αντικειμενικό (εφικτό) στόχο και κάνουν την ανανέωση χρησιμοποιώντας την αμοιβή r_{t+1} και τον υπολογισμό $V(s_{t+1})$.

Οι TD μέθοδοι δεν χρειάζεται να περιμένουν την τελική έξοδο για να αρχίσουν τη μάθηση, αλλά ανανεώνουν τους υπολογισμούς τους βασιζόμενοι εν μέρει σε άλλους υπολογισμούς που είναι ήδη γνωστοί. Για το λόγο αυτό χαρακτηρίζονται μέθοδοι αυτοδύναμης εκκίνησης *(bootstrapping methods),* όπως και οι μέθοδοι Δυναμικού Προγραμματισμού.

Συνοψίζοντας, λοιπόν, οι TD μέθοδοι συνδυάζουν την δειγματοληψία των Monte Carlo μεθόδων με την αυτοδύναμη εκκίνηση *(bootstrapping)* των μεθόδων Δυναμικού Προγραμματισμού αποκτώντας πλεονεκτήματα και των δύο μεθόδων.

3.2. Παράδειγμα: Οδηγώντας προς το σπίτι

Τελειώνοντας τη δουλειά και οδηγώντας προς το σπίτι προσπαθείς να προβλέψεις πόση ώρα θα χρειαστείς για να φτάσεις. Έστω π.χ., την Παρασκευή φεύγεις από το γραφείο ακριβώς στις 6 και υπολογίζεις ότι θα σου πάρει 30 λεπτά να φτάσεις στο σπίτι. Μπαίνεις στο αυτοκίνητό σου στις 6:05 όταν ξαφνικά αρχίζει να βρέχει. Η βροχή επηρεάζει την κίνηση οπότε υπολογίζεις ότι θα χρειαστείς 35 λεπτά από τότε για να φτάσεις στο σπίτι, συνολικά δηλαδή 40 λεπτά. 15 λεπτά αργότερα έχεις διασχίσει τη λεωφόρο σε πολύ καλό χρόνο. Βγαίνοντας σε έναν δευτερεύοντα δρόμο υπολογίζεις εκ νέου το συνολικό χρόνο και καταλήγεις στο συμπέρασμα ότι θα χρειαστείς συνολικά 35 λεπτά. Δυστυχώς, στο σημείο αυτό κολλάς πίσω από ένα φορτηγό και δεν μπορείς να το προσπεράσεις λόγω της στενότητας του δρόμου. Αναγκαστικά ακολουθείς το φορτηγό έως ότου στρίψεις στη διασταύρωση προς το σπίτι σου, εκείνη τη στιγμή η ώρα είναι 6:40. Τρία λεπτά αργότερα είσαι σπίτι.

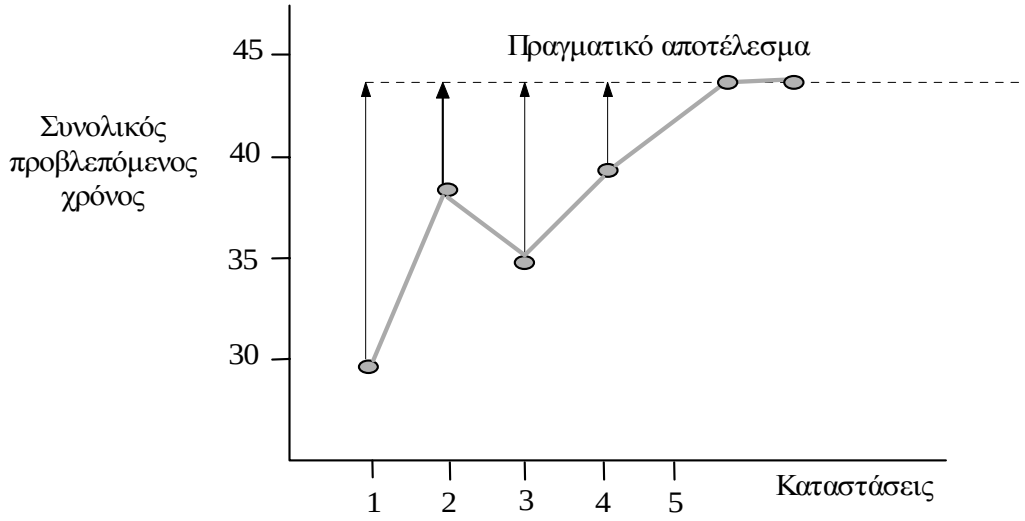
Η ακολουθία των καταστάσεων, του χρόνου και των προβλέψεων του παραδείγματος δίνεται στον ακόλουθο πίνακα:

#	Κατάσταση	Απαιτούμενος χρόνος	Υπόλοιπος προβλεπόμενος χρόνος	Συνολικός προβλεπόμενος χρόνος
1	Φεύγω από το γραφείο Παρασκευή στις 6	0 *λεπτά*	30 *λεπτά*	30
2	Αυτοκίνητο – Βροχή	5 *λεπτά*	35 *λεπτά*	40 *λεπτά*
3	Τέλος αυτοκινητοδρόμου	20 *λεπτά*	15 *λεπτά*	35 *λεπτά*
4	Φορτηγό	30 *λεπτά*	10 *λεπτά*	40 *λεπτά*
5	Δρόμος προς το σπίτι	40 *λεπτά*	3 *λεπτά*	43 *λεπτά*
6	Σπίτι	43 *λεπτά*	0 *λεπτά*	43 *λεπτά*

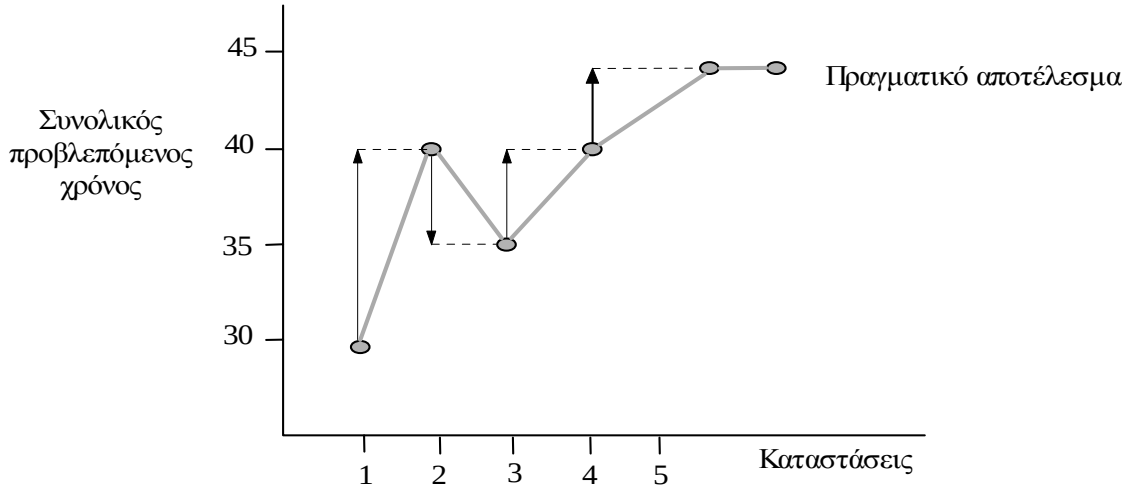
Πίνακας 3.1 Παράδειγμα πρόβλεψης μέσω TD – μεθόδου

Σύμφωνα με τις Monte Carlo μεθόδους η μάθηση θα μπορούσε να αρχίσει μόνο στο τέλος, όταν δηλαδή θα ήμασταν στο σπίτι. Αντιθέτως, στην περίπτωση των TD μεθόδων δεν χρειάζεται να περιμένουμε τόσο πολύ, μπορούμε να ανανεώσουμε τον υπολογισμό μας την αμέσως επόμενη χρονική στιγμή (Δεν χρειάζεται δηλαδή να φτάσουμε σπίτι για να συνειδητοποιήσουμε ότι ο αρχικός μας υπολογισμός δεν ισχύει πλέον. Βλέποντας καθοδόν ότι τα πράγματα δεν ακολουθούν τις αρχικές μας προβλέψεις αναθεωρούμε την αρχική μας εκτίμηση και δεν περιμένουμε να είμαστε στο σπίτι σε 30 λεπτά).

Στο ακόλουθο σχήμα φαίνονται οι αλλαγές έτσι όπως προτάθηκαν από τις μεθόδους Monte Carlo (Σχήμα 3.1) και από τις TD μεθόδους (Σχήμα 3.2).



Σχήμα 3.1 Οι αλλαγές που προτάθηκαν από τις μεθόδους Monte Carlo



Σχήμα 3.2 Οι αλλαγές που προτάθηκαν από τις TD μεθόδους.

3.3. Πλεονεκτήματα της Μάθησης Χρονικών Διαφορών (TD learning)

Οι μέθοδοι μάθησης χρονικών διαφορών *(TD learning methods)* είναι αυξητικές *(incremental)*, με αποτέλεσμα η πολυπλοκότητα υπολογισμού τους να είναι μικρή. Η μάθηση συντελείται μεταξύ διαδοχικών χρονικών στιγμών και όχι στο τέλος, όπως συμβαίνει με τις Monte Carlo μεθόδους.

Επιπλέον, οι TD μέθοδοι χρησιμοποιούν πιο αποδοτικά την εμπειρία τους με αποτέλεσμα να συγκλίνουν ταχύτερα και να κάνουν καλύτερες προβλέψεις.

Επίσης δεν χρειάζεται να γνωρίζουν το μοντέλο του περιβάλλοντος, όπως οι μέθοδοι Δυναμικού Προγραμματισμού. Μαθαίνουν κατευθείαν από την εμπειρία που έχουν σχετικά με μια συγκεκριμένη στρατηγική π και ανανεώνουν σε κάθε χρονική στιγμή τον υπολογισμό τους V για την V^{π} .

Στην πραγματικότητα οι TD μέθοδοι δεν χρησιμοποιούνται μόνο για τον υπολογισμό συναρτήσεων προσέγγισης. Αποτελούν γενικότερες μεθόδους για την μάθηση της δυνατότητας μακροπρόθεσμων προβλέψεων σε δυναμικά συστήματα. Για παράδειγμα μπορούν να χρησιμοποιηθούν για την πρόβλεψη οικονομικών δεδομένων, καιρικών προτύπων, τάσεων αγοράς, συμπεριφοράς ζώων, αναγνώριση φωνής... Και παρά το γεγονός ότι οι εφαρμογές τους είναι ήδη αρκετές αξίζει να σημειώσουμε πως οι δυνατότητες και η δυναμική των TD μεθόδων δεν έχει εξερευνηθεί ακόμη πλήρως.

3.4. Μάθηση Χρονικών Διαφορών TD(λ)

Μέχρι στιγμής δεν έχουμε αναφερθεί ουσιαστικά στο γεγονός ότι οι μέθοδοι TD learning αποτελούν μια μορφή καθυστερημένης ενισχυτικής μάθησης. Αντιθέτως, έχουμε υποθέσει πως όλες οι εκτιμήσεις είναι ισοδύναμες και δεν εξαρτώνται από τη χρονική στιγμή κατά την οποία πραγματοποιήθηκαν. Το σωστό θα ήταν να λάβουμε υπόψη και το χρονικό παράγοντα. Ακριβώς αυτό κάνει ο παράγοντας παράληψης *(forgetting factor) λ,* μία ευριστική παράμετρος με τιμές $0 \le \lambda \le 1$. Το *λ* καθορίζει το ρυθμό μείωσης της ανάθεσης πίστωσης *(credit assignment)* σε μια ενέργεια, δηλαδή καθορίζει κατά πόσο ευθύνονται οι προηγούμενες σωστές εκτιμήσεις για ένα λάθος που συμβαίνει σε μία δεδομένη μεταγενέστερη χρονική στιγμή. Έτσι, πολλαπλασιάζουμε τις εκτιμήσεις που συνέβησαν τη χρονική στιγμή *k* με έναν παράγοντα βαρύτητας λ^k και η ανανέωση έχει πλέον τη μορφή:

$$\Delta w_t = \alpha(V_{t+1} - V_t)\sum_{k=1}^{t} \lambda^{t-k} \nabla_w V_k$$

όπου

α: ο ρυθμός μάθησης *(learning rate)*

V_t: η πρόβλεψη τη χρονική στιγμή *t*

$\nabla_w V_k$: το διάνυσμα τελεστής *(gradient vector)* της πρόβλεψης ως προς το διάνυσμα των βαρών του δικτύου τη χρονική στιγμή *t*

λ: ο παράγοντας παράληψης *(forgetting factor)*

Ας εξετάσουμε τι συμβαίνει για τις διάφορες τιμές του *λ*:

- Για *λ=0,* η απόδοση πίστωσης σε μια ενέργεια μειώνεται γρήγορα και στην ουσία μόνο η τελευταία κατάσταση επηρεάζεται. Έτσι το λάθος που μπορεί να συμβεί μια δεδομένη χρονική στιγμή δεν μεταφέρεται σε εκτιμήσεις προγενέστερων χρονικών στιγμών.
- Για *λ=1,* η απόδοση πίστωσης σε μια ενέργεια μειώνεται αργά και επηρεάζεται σχεδόν ολόκληρη η ακολουθία καταστάσεων. Έτσι το λάθος που μπορεί να συμβεί μια δεδομένη χρονική μεταφέρεται αναλλοίωτο σχεδόν στις προηγούμενες εκτιμήσεις προκειμένου να τις διορθώσει.

Μια κατάλληλη τιμή του *λ* επιτρέπει πιο γρήγορη σύγκλιση. Δεδομένου δε και του ρυθμού μάθησης *α,* αν ισχύει:

$$\sum_K \alpha_k = \infty \ , \ \sum_K \alpha^2_k < \infty$$

μπορούμε να είμαστε σίγουροι για τη σύγκλιση.

3.5. Βελτιστοποίηση και σύγκλιση

Έστω ότι η εμπειρία που διαθέτουμε είναι πεπερασμένη, δηλαδή μιλάμε για τις περιπτώσεις Μαρκοβιανών διαδικασιών. Μια κοινή αντιμετώπιση μέσω των μεθόδων TD μάθησης θα ήταν να χρησιμοποιήσουμε επανειλημμένα την ίδια εμπειρία μέχρι η μέθοδος να συγκλίνει σε μία απάντηση. Αν *V* είναι η συνάρτηση προσέγγισης, οι αυξήσεις υπολογίζονται σε κάθε χρονική στιγμή *t* που επισκεπτόμαστε μη τελικές καταστάσεις, αλλά η συνάρτηση *V* αλλάζει μόνο μία φορά με το άθροισμα των επιμέρους αυξήσεων. Τότε η διαθέσιμη εμπειρία συνδυάζεται με τη νέα συνάρτηση προσέγγισης και παράγουν μία νέα ολική αύξηση. Η διαδικασία αυτή επαναλαμβάνεται μέχρι τη σύγκλιση και ονομάζεται ανανέωση παρτίδας *(batch updating),* επειδή ακριβώς οι ανανεώσεις πραγματοποιούνται μετά την επεξεργασία κάθε παρτίδας εκπαιδευτικών παραδειγμάτων.

Η έννοια της επανάληψης που αναφέραμε πριν δικαιολογεί και το γεγονός ότι οι μέθοδοι TD learning έχουν, στην πράξη τουλάχιστον, μεγαλύτερους ρυθμούς μάθησης. Από τη στιγμή που αποσκοπούν σε ένα καλύτερο τελικό αποτέλεσμα είναι προφανές ότι και μετά από κάθε βήμα θα παρουσιάζονται βελτιωμένες.

Τίθεται πλέον το ερώτημα για το αν οι TD(λ) μέθοδοι συγκλίνουν. Το 1988 ο Sutton [Sutton 1988] απέδειξε ότι στην περίπτωση Μαρκοβιανών διαδικασιών οι αλγόριθμοι TD(0), TD(1) συγκλίνουν στις σωστές τιμές. Ο Dayan [1992] επέκτεινε τη θεωρία του Sutton για $0 \le \lambda \le 1$. Στην πράξη, η γενικότητα του TD(λ) επαληθεύτηκε από τον Tesauro [Tesauro 1995], ο οποίος χρησιμοποίησε το 1992 τον αλγόριθμο TD(λ) με *λ=0* για να εκπαιδεύσει έναν agent στο τάβλι. Σήμερα, έχοντας παίξει χιλιάδες παιχνίδια το TD-Gammon του Tesauro θεωρείται πλέον ικανό να ανταγωνιστεί με επιτυχία παγκόσμιους πρωταθλητές.

Στην πράξη μάλιστα, σε περιπτώσεις Μαρκοβιανών διαδικασιών έχει παρατηρηθεί οι TD μέθοδοι να συγκλίνουν πιο γρήγορα από τις μεθόδους Monte Carlo.

3.6. Sarsa learning: On policy TD control

Ο αλγόριθμος του SARSA αποτελεί μια TD μέθοδο που μαθαίνει τις συναρτήσεις αξιολόγησης κινήσεων βάσει ενός μηχανισμού αυτοδύναμης εκκίνησης *(bootstrapping mechanism),* αυτό σημαίνει πως οι εκτιμήσεις που κάνει στηρίζονται σε προηγούμενες εκτιμήσεις.

Σε κάθε βήμα, ο αλγόριθμος SARSA ανανεώνει τις εκτιμήσεις των συναρτήσεων αξιολόγησης *Q(s,a)* χρησιμοποιώντας την πεντάδα *(s,a,r,s',a'),* η οποία εξάλλου έδωσε και το όνομα του αλγορίθμου.

Ο ψευδοκώδικας του αλγορίθμου SARSA είναι ο ακόλουθος:

Αλγόριθμος Sarsa

1. Αρχικοποίησε την συνάρτηση αξιολόγησης *Q(s,a)* με τυχαίες τιμές.
2. Επανέλαβε για κάθε επεισόδιο:

   Έστω η κατάσταση *s*

   Επέλεξε κίνηση *a* (δοθείσας της *s)* χρησιμοποιώντας την στρατηγική *(policy)* που προκύπτει από το Q (π. χ, ε-greedy)

   Επανέλαβε (για κάθε βήμα του επεισοδίου):

   Επέλεξε την κίνηση α, παρατήρησε τις τιμές *r,s'*.

   $$Q(s,a) \leftarrow Q(s,a) + a[r + Q(s',a') - Q(s,a)]$$
   $$s \leftarrow s'; a \leftarrow a'$$

   μέχρι η s να είναι τελική κατάσταση
3. Επανέλαβε το βήμα 2 για το σύνολο των επεισοδίων.

Επειδή ακριβώς πρόκειται για μία μέθοδο βήμα προς βήμα μάθησης *(step by step learning)*, δεν υφίσταται το πρόβλημα της προτίμησης μιας στρατηγικής που οδηγεί τον agent στην ίδια πάντα κατάσταση. Ο agent μαθαίνει γρήγορα (κατά τη διάρκεια ενός επεισοδίου) ότι τέτοιες στρατηγικές είναι φτωχές και συνεπώς πρέπει να τις αποφεύγει αναζητώντας καλύτερες.

Όσον αφορά τη σύγκλιση, ο αλγόριθμος Sarsa συγκλίνει με πιθανότητα 1 σε μία βέλτιστη στρατηγική αν η συχνότητα επίσκεψης σε όλα τα ζεύγη καταστάσεων-κινήσεων είναι πολύ μεγάλη.

3.7. Q-Learning: Off policy TD control

Ένας από τους πιο σημαντικούς σταθμούς στην ενισχυτική μάθηση ήταν και η ανάπτυξη ενός off-policy TD ελέγχου αλγορίθμου, γνωστού ως Q-learning (Watkins, 1989). Η συνάρτηση αξιολόγησης κινήσεων *Q* που μαθαίνει τελικά ο agent, προσεγγίζει την βέλτιστη συνάρτηση αξιολόγησης κινήσεων *Q** ανεξάρτητα από την στρατηγική που ακολουθείται. Το γεγονός αυτό διευκολύνει την ανάλυση του αλγορίθμου και οδηγεί σε πιο γρήγορη σύγκλιση. Ωστόσο στην περίπτωση αυτή η στρατηγική έχει νόημα καθώς καθορίζει ποια ζεύγη καταστάσεων-κινήσεων επισκεπτόμαστε και ανανεώνουμε.

Ο ψευδοκώδικας του αλγορίθμου SARSA είναι ο ακόλουθος:

Αλγόριθμος Q-learning

1. Αρχικοποίησε την συνάρτηση αξιολόγησης *Q(s,a)* με τυχαίες τιμές.
2. Επανέλαβε για κάθε επεισόδιο:

   Έστω η κατάσταση *s*

   Επανέλαβε για κάθε βήμα του επεισοδίου:

   Επέλεξε κίνηση *α* (δοθείσας της *s)* χρησιμοποιώντας την στρατηγική *(policy)* που προκύπτει από το Q (π. χ, ε-greedy)

   Επέλεξε την κίνηση α, παρατήρησε τις τιμές *r,s'*.

   $Q(s,a) \leftarrow Q(s,a) + a[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$

   $s \leftarrow s'$

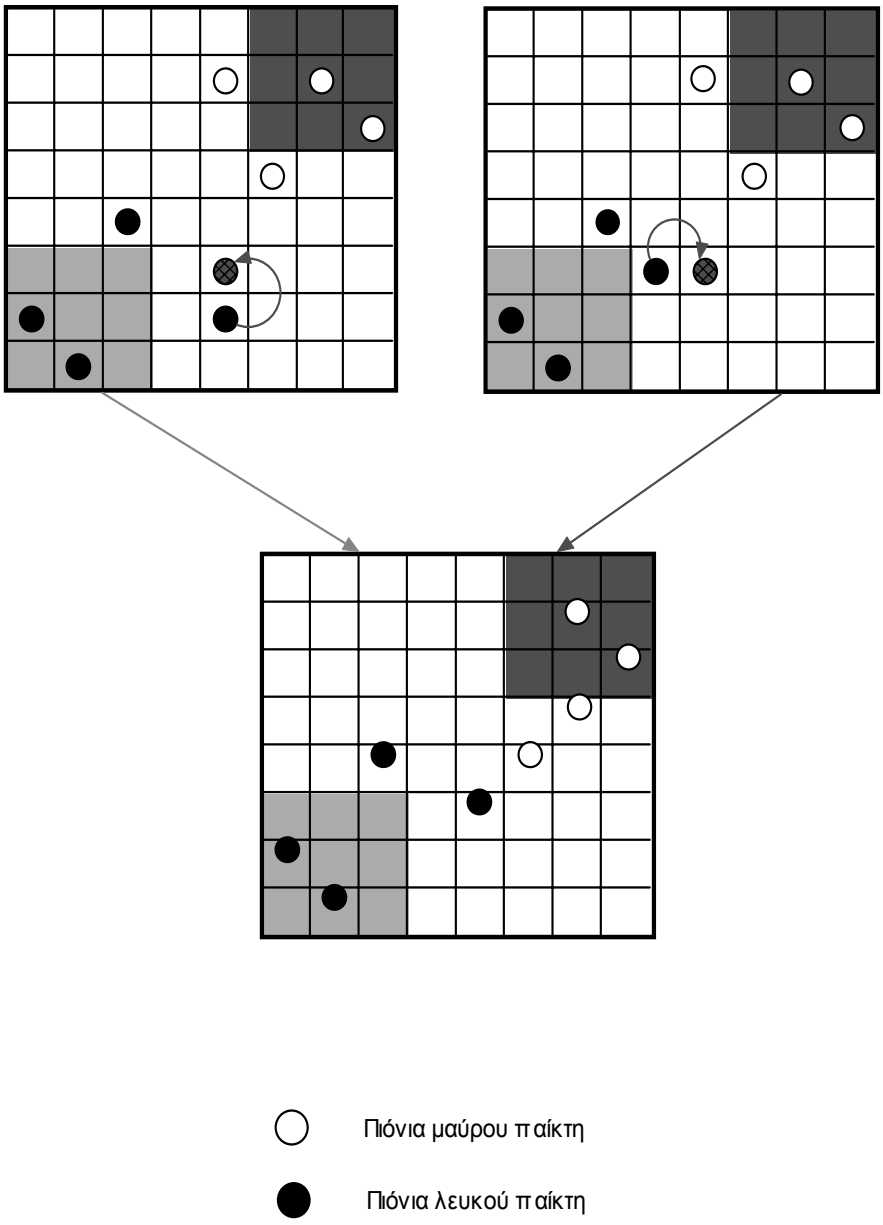
   μέχρι η s να είναι τελική κατάσταση
3. Επανέλαβε το βήμα 2 για ένα σύνολο επεισοδίων.

3.8. Μετά - καταστάσεις *(after states)* στα παιχνίδια

Με τον όρο μετά - κατάσταση *(after state*) καλούμε τη διαμόρφωση της σκακιέρας μετά την κίνηση του παίκτη *(agent)*. Οι συναρτήσεις αποτίμησης αυτών των καταστάσεων καλούνται συναρτήσεις αποτίμησης μετά-καταστάσεων (*after states value functions*). Οι καταστάσεις αυτές εμφανίζονται συνήθως στα παιχνίδια, όπου τυπικά γνωρίζουμε την άμεση επίδραση των κινήσεών μας. Για παράδειγμα, στο παιχνίδι μας, ξέρουμε για κάθε πιθανή μας κίνηση ποια θα είναι η διαμόρφωση της σκακιέρας, αλλά δεν ξέρουμε την κίνηση του αντιπάλου. Ωστόσο, μπορούμε να εκμεταλλευτούμε αυτού του είδους την πληροφορία μέσω των συναρτήσεων αποτίμησης μετά - καταστάσεων που «παράγουν» μ' αυτό τον τρόπο μια πιο αποτελεσματική και γρήγορη μέθοδο μάθησης.

Ο λόγος για τον οποίο η μάθηση είναι πιο αποτελεσματική είναι προφανής: Από πολλές καταστάσεις ακολουθώντας διαφορετικές κινήσεις μπορούμε να καταλήξουμε στην ίδια κατάσταση, όπως φαίνεται στο επόμενο σχήμα (Σχήμα 3.3).

Στις περιπτώσεις μετά-καταστάσεων παρόλο που τα ζεύγη καταστάσεων - κινήσεων είναι διαφορετικά, «παράγουν» την ίδια μετά - κατάσταση και για το λόγο αυτό η συνάρτηση αποτίμησης θα πρέπει να τους αποδώσει την ίδια τιμή. Στο παιχνίδι μας (Σχήμα 3.3) π.χ. οποιαδήποτε γνώση για το ζεύγος κατάστασης-κίνησης αριστερά θα μεταφερθεί αυτόματα και στο ζεύγος κατάστασης-κίνησης δεξιά, μειώνοντας έτσι το χρόνο μάθησης.



Σχήμα 3.3 Παράδειγμα μετά-καταστάσεων

4. Ίχνη καταλληλότητας *(Eligibility traces)*

4.1. Εισαγωγή

Δύο είναι οι βασικοί μηχανισμοί που χρησιμοποιούνται στον τομέα της ενισχυτικής μάθησης για την αντιμετώπιση του προβλήματος καθυστερημένης αμοιβής *(delay reward problem).* Ο ένας είναι η μάθηση χρονικών διαφορών *(temporal difference learning – TD)* στην οποία έχουμε ήδη αναφερθεί. Όπως είδαμε, η TD μάθηση κατασκευάζει εσωτερικές αμοιβές *(rewards)* που καθυστερούν λιγότερο από τις πραγματικές εξωτερικές. Το μειονέκτημα όμως των TD μεθόδων είναι ότι για να εξαλείψουν πλήρως την καθυστέρηση θα πρέπει οι διαδικασίες να είναι Μαρκοβιανές, γεγονός που δύσκολα ισχύει στην πράξη. Συνήθως, υπάρχει πάντα κάποια καθυστέρηση μεταξύ μιας πράξης και της απόδοσης πίστωσης σ' αυτή, και σχεδόν πάντα υπάρχει καθυστέρηση πριν ολοκληρωθεί η μάθηση.

Ο δεύτερος μηχανισμός είναι τα ίχνη καταλληλότητας *(eligibility traces).* Χρησιμοποιήθηκαν για πρώτη φορά στον τομέα της Ενισχυτικής Μάθησης το 1972 από τον Klopf και από τότε χρησιμοποιούνται σε πολλά RL συστήματα. Η ιδέα είναι απλή: Κάθε φορά που επισκεπτόμαστε μια κατάσταση *s* κρατάμε ένα προσωρινό αρχείο, ένα ίχνος *(trace)*, που εξασθενεί σταδιακά στα επόμενα βήματα. Στο ίχνος αυτό σημειώνουμε ότι έχουμε επισκεφτεί την κατάσταση *s,* η οποία ονομάζεται πλέον κατάλληλη *(eligible)*. Η πίστωση που θα αποδοθεί στην κατάσταση *s* εξαρτάται από το αν η επόμενη κατάσταση στην οποία θα μεταβούμε είναι καλή ή κακή.

Τα ίχνη καταλληλότητας μπορούν να συνδυαστούν με σχεδόν όλες τις TD μεθόδους μάθησης π.χ. τον αλγόριθμο TD(λ), την Q-μάθηση και τον αλγόριθμο Sarsa.

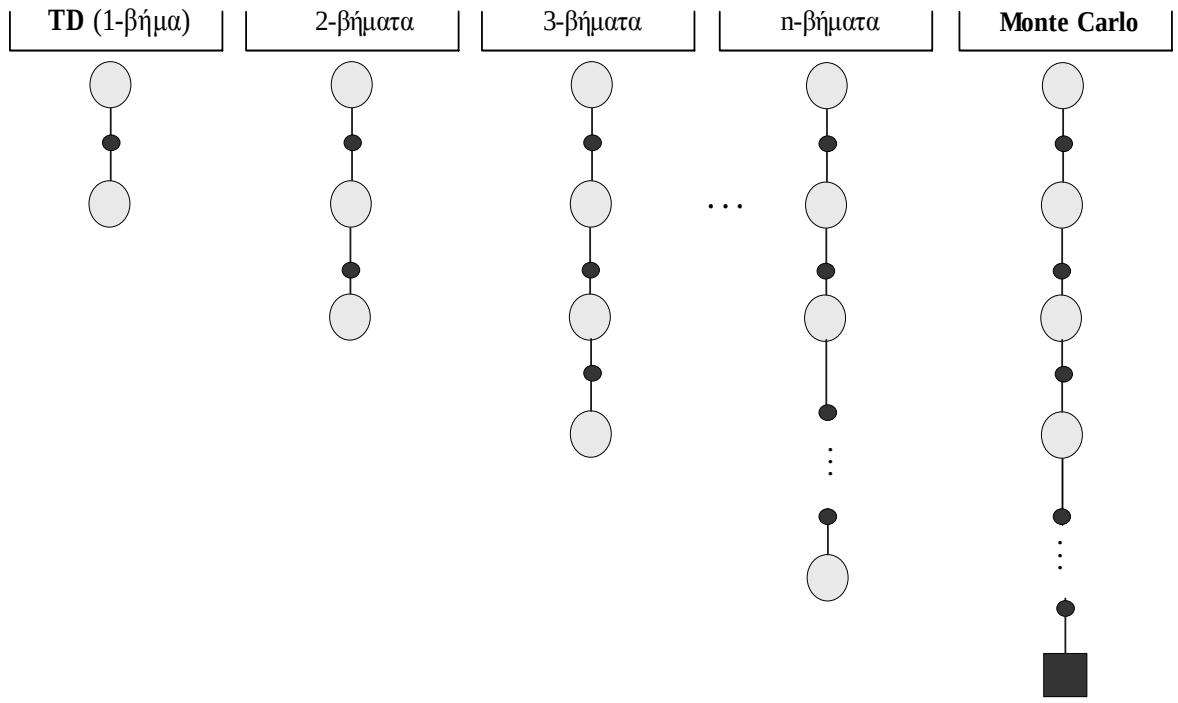
4.2. TD (λ) και Monte Carlo μέθοδοι

Όταν οι TD μέθοδοι συνδυάζονται με τα ίχνη καταλληλότητας, παράγουν μια οικογένεια μεθόδων με χαρακτηριστικά TD μεθόδων μάθησης και Monte Carlo μεθόδων. Για το λόγο αυτό εξάλλου, θεωρείται ότι τα ίχνη καταλληλότητας μειώνουν το χάσμα μεταξύ των μεθόδων TD και Monte Carlo. Η προσέγγιση αυτή όσον αφορά στα ίχνη καταλληλότητας είναι θεωρητική και ονομάζεται «προσέγγιση προς τα εμπρός» *(forward view).*

Υπάρχει και μία άλλη προσέγγιση που αντιμετωπίζει τα ίχνη καταλληλότητας πιο πρακτικά και ονομάζεται «προσέγγιση προς τα πίσω» *(backward view)*. Βάσει αυτής τα ίχνη καταλληλότητας «αποτελούν» ένα είδος εγγραφής στην οποία αποθηκεύεται η συχνότητα εμφάνισης ενός γεγονότος, π.χ. πόσες φορές επισκεφτήκαμε μία συγκεκριμένη κατάσταση *s*. Η κατάσταση *s* ονομάζεται κατάλληλη *(eligible)* και της αποδίδεται πίστωση ανάλογα με το πόσο καλή ή κακή είναι η κατάσταση που τη διαδέχεται.

Τονίζουμε ωστόσο πως και οι δύο προσεγγίσεις όσον αφορά στα ίχνη καταλληλότητας είναι ισοδύναμες [Sutton & Barto 1998].

Ας θεωρήσουμε το πρόβλημα του υπολογισμού της συνάρτησης αξιολόγησης V^{π} από ένα σύνολο παραδειγμάτων βάσει μιας τακτικής *π.* Οι Monte Carlo μέθοδοι θα δημιουργούσαν ένα διάγραμμα ενημέρωσης *(backup diagram)*, στο Κεφάλαιο 2 έχουμε ορίσει την έννοια των διαγραμμάτων ενημέρωσης, για κάθε κατάσταση λαμβάνοντας υπόψη τις πιστώσεις από την συγκεκριμένη κατάσταση και μέχρι το τέλος του επεισοδίου. Οι TD μέθοδοι από την άλλη θα δημιουργούσαν το διάγραμμα ενημέρωσης χρησιμοποιώντας κάποιες από τις αμέσως επόμενες πιστώσεις, αλλά όχι απαραίτητα όλες. Οι μέθοδοι αυτοί είναι γνωστοί ως TD μέθοδοι n-βημάτων *(n-step TD methods)* (Σχήμα 4.1).



Σχήμα 4.1 Από τα 1-,2-,3-,...n-βημάτων backup των TD μεθόδων στα n-βημάτων backup των Monte Carlo μεθόδων. *(Το σχήμα προσαρμόστηκε από το [Sutton & Barto 1998],Κεφάλαιο 7, Παράγραφος 7.1, Σχήμα 7.1)*

4.3. TD (λ): προσέγγιση προς τα εμπρός *(forward view)*

Τα διαγράμματα ενημέρωσης*(backup diagram)* (Σχήμα 4.1) μπορούν να δημιουργηθούν όχι μόνο μετά από κάποιο βήμα *n* αλλά και μετά το μέσο όρο κάποιων βημάτων. Ο αλγόριθμος TD (λ) μπορεί να θεωρηθεί ως ένας τρόπος υπολογισμού του μέσου όρου *n*- διαγραμμάτων ενημέρωσης. Ο μέσος όρος περιέχει όλα τα διαγράμματα ενημέρωσης *n*-βημάτων καθένα πολλαπλασιασμένο με έναν παράγοντα λ^{n-1}, $0 \leq \lambda \leq 1$. Το διάγραμμα ενημέρωσης που προκύπτει καλείται *λ*- επιστροφή *(λ- return)* και δίνεται από τη

$$R_t^{\lambda} = (1 - \lambda) * \sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)}$$

σχέση:

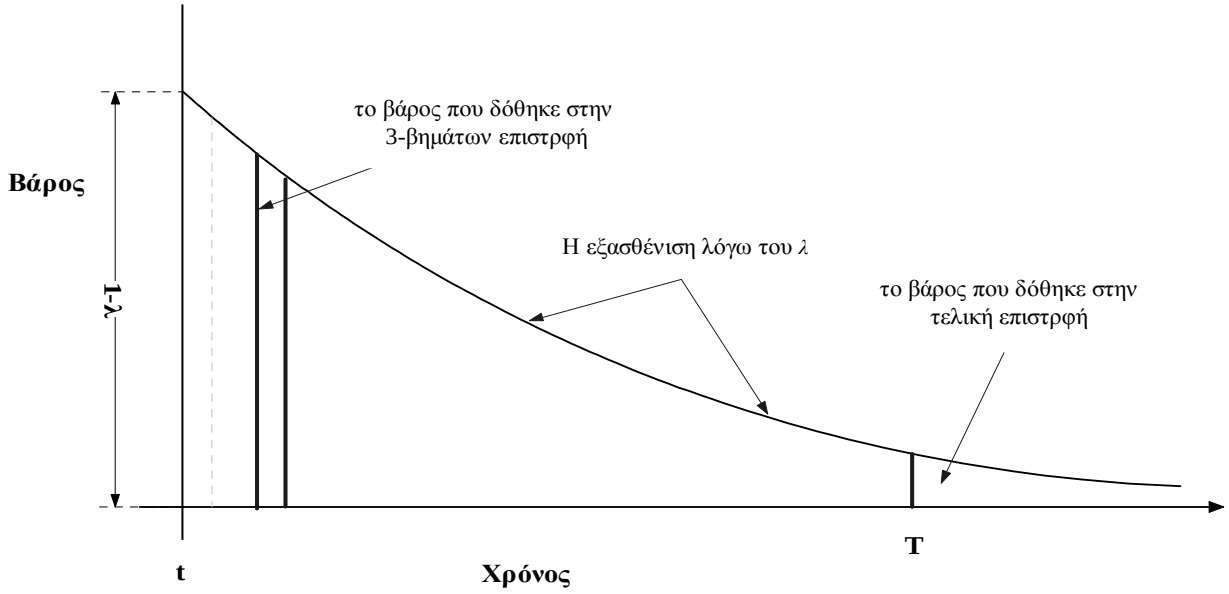
$$R_t^{\lambda} = (1-\lambda)R_t^{(1)} + (1-\lambda)\lambda R_t^{(2)} + \cdots + (1-\lambda)\lambda^{n-1} R_t^{(n)} + \lambda^{N-1} R_t^{(N)}$$

όπου

Ο παράγοντας $R_t^{(n)}$ καλείται επιστροφή *n*- βημάτων τη χρονική στιγμή *t (n-step return)*.

Ο παράγοντας *(1-λ)* εξασφαλίζει ότι το άθροισμα των βαρών είναι *1*.

Όσον αφορά στην ακολουθία των βαρών (Σχήμα 4.2) η επιστροφή *1*- βήματος έχει το μεγαλύτερο βάρος *(1-λ),* ακολουθεί η επιστροφή *2*- βημάτων με βάρος $(1-\lambda)\lambda^2$ κ. ο. κ. Το βάρος ξεθωριάζει κατά *λ* σε κάθε βήμα. Όταν φτάσουμε σε μία τελική κατάσταση, όλες οι επιστροφές *n*- βημάτων ισούνται με R_t.



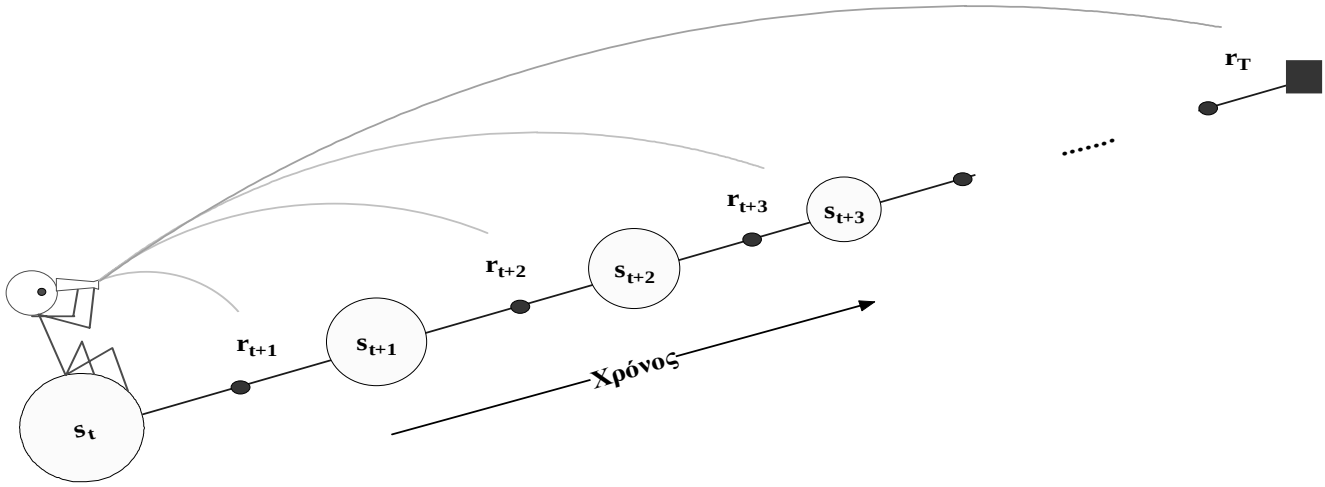
Σχήμα 4.2 Η ακολουθία των βαρών στην λ-επιστροφή σε κάθε ένα από τα n-βήματα. (Το σχήμα προσαρμόστηκε από το [Sutton & Barto 1998],Κεφάλαιο 7, Παράγραφος 7.2, Σχήμα 7.3)

Ο αλγόριθμος που χρησιμοποιεί τη *λ*- επιστροφή για τη δημιουργία των διαγραμμάτων ενημέρωσης καλείται αλγόριθμος *λ*- επιστροφής *(λ- return algorithm)*. Σε κάθε βήμα υπολογίζει μία αύξηση $\Delta V_t(s_t)$ στην αξία της κατάστασης στην οποία βρίσκεται το σύστημα τη συγκεκριμένη χρονική στιγμή βάσει της σχέσης:

$$\Delta V_t(s_t) = a\left[R_t^{\lambda} - V_t(s_t)\right]$$

Όσον αφορά στις άλλες καταστάσεις η αύξηση για τη συγκεκριμένη χρονική στιγμή είναι *0*.

Η προσέγγιση αυτή ονομάζεται «προσέγγιση προς τα εμπρός» *(forward view)* επειδή για κάθε κατάσταση που επισκεπτόμαστε κοιτάμε τις αντίστοιχες μελλοντικές αμοιβές και καταστάσεις και αποφασίζουμε πως θα τα συνδυάσουμε καλύτερα (Σχήμα 4.3). Αφού ανανεώσουμε τη συγκεκριμένη κατάσταση δεν χρειάζεται να ασχοληθούμε ξανά μαζί της. Συνεχίζουμε επαναλαμβάνοντας την ίδια διαδικασία για την επόμενη κατάσταση κ. ο. κ.



Σχήμα 4.3 Η προς τα εμπρός προσέγγιση του TD(λ). *(Το σχήμα προσαρμόστηκε από το [Sutton & Barto 1998],Κεφάλαιο 7, Παράγραφος 7.2, Σχήμα 7.5)*

4.4. TD (λ): προσέγγιση προς τα πίσω *(backward view)*

Η «προσέγγιση προς τα εμπρός» που είδαμε μόλις πριν είναι δύσκολα υλοποιήσιμη καθώς απαιτεί σε κάθε βήμα γνώση του τι θα γίνει στα επόμενα βήματα. Το πρόβλημα αυτό έρχεται να επιλύσει η «προσέγγιση προς τα πίσω» *(backward view)*.

Βάσει αυτής το ίχνος καταλληλότητας για μια κατάσταση *s* τη χρονική στιγμή *t* δίνεται από τη σχέση:

$$e_t(s) = \begin{cases} \gamma*\lambda*e_{t-1}(s)+1, & \text{αν } s=s_t \\ \gamma*\lambda*e_{t-1}(s), & \text{διαφορετικά} \end{cases}$$

όπου

$e_t(s)$*:* το ίχνος για την κατάσταση *s* τη χρονική στιγμή *t*.

λ: ο παράγοντας παράληψης *(forgetting factor)* με τιμές $0 \le \lambda \le 1$

γ: η παράμετρος του ρυθμού μείωσης, μια σταθερά με τιμές $0 \le \gamma < 1$ που είναι γνωστή ως παράμετρος εξασθένισης του ίχνους *(trace decay parameter).* Το ίχνος καταλληλότητας στην περίπτωση αυτή ονομάζεται ίχνος συσσώρευσης *(accumulating trace)*, επειδή ακριβώς αυξάνεται κάθε φορά που επισκεπτόμαστε μία κατάσταση και εξασθενεί σταδιακά στη συνέχεια όταν δεν επισκεπτόμαστε τη συγκεκριμένη κατάσταση (Σχήμα 4.5).

Σε κάθε βήμα τα ίχνη καταλληλότητας καταγράφουν ποιες καταστάσεις έχουμε επισκεφτεί πιο πρόσφατα, όπου το πρόσφατα καθορίζεται από τον παράγοντα *γ*λ.* Μεγαλύτερη πίστωση δίνεται στις πιο πρόσφατες και στις πιο συχνά εμφανιζόμενες καταστάσεις (Sutton 1984). Τα ίχνη καταλληλότητας δείχνουν κατά κάποιο τρόπο το βαθμό στον οποίο μια κατάσταση είναι κατάλληλη *(eligible)* για να της εφαρμοστούν αλλαγές όσον αφορά στη μάθηση.

Η ανανέωση γίνεται βάσει της σχέσης:

$$\Delta V_t(s) = a * \delta_t * e_t(s), \ \forall s \in S$$

Στην περίπτωση που οι ανανεώσεις αυτές γίνονται σε κάθε βήμα (on-line) και όχι στο τέλος του επεισοδίου *(off-line)* ο ψευδοκώδικας θα μπορούσε να είναι της ακόλουθης μορφής:

On – line TD (λ)

Αρχικοποίησε την συνάρτηση αξιολόγησης *V(s)* με τυχαίες τιμές και θέσε *e(s)=0,* $\forall s \in S$.

Επανέλαβε για κάθε επεισόδιο:

Αρχικοποίησε την κατάσταση s.

Επανέλαβε για κάθε βήμα του επεισοδίου:

Επέλεξε κίνηση *α* (δοθείσας της s*)* χρησιμοποιώντας την στρατηγική *(policy)* π.

Εκτέλεσε την κίνηση *α*, παρατήρησε την αμοιβή *(reward) r* και την επόμενη κατάσταση *s'*.

$$\delta \leftarrow r + \gamma * V(s') - V(s)$$

$$e(s) \leftarrow e(s) + 1$$

Για όλα τα s:

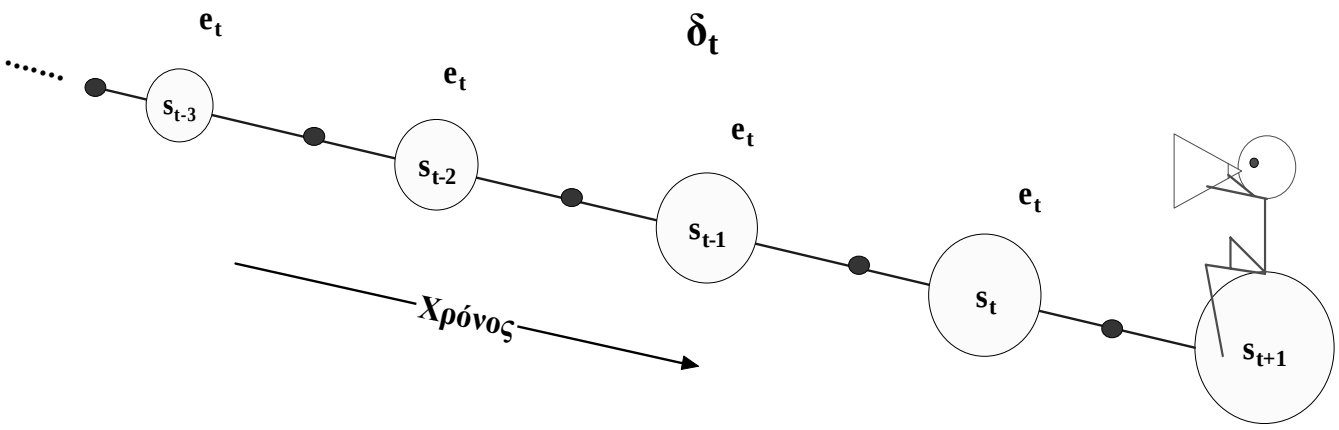
$$V(s) \leftarrow V(s) + \alpha * \delta * e(s)$$

$$e(s) \leftarrow \gamma * \lambda * e(s)$$

$$s \leftarrow s'$$

μέχρι η s να είναι τελική κατάσταση.

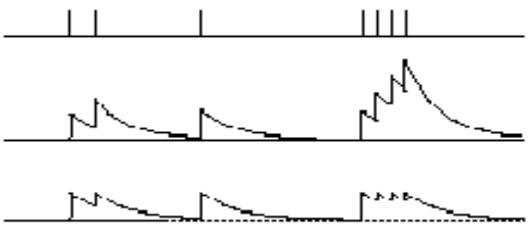
Στην «προς τα πίσω προσέγγιση» του TD(λ) είμαστε προσανατολισμένοι στο παρελθόν. Σε κάθε χρονική στιγμή ελέγχουμε το τρέχον TD-λάθος και το σχετίζουμε με όλες τις προηγούμενες καταστάσεις ανάλογα με το ίχνος καταλληλότητας της κάθε κατάστασης τη συγκεκριμένη χρονική στιγμή. Οποιαδήποτε αλλαγή δηλαδή εξαρτάται από το τρέχον TD-λάθος σε συνδυασμό με τα ίχνη καταλληλότητας προηγούμενων καταστάσεων (Σχήμα 4.4).



Σχήμα 4.4 Η προς τα πίσω προσέγγιση του TD(λ). *(Το σχήμα προσαρμόστηκε από το [Sutton & Barto 1998],Κεφάλαιο 7, Παράγραφος 7.3, Σχήμα 7.8)*

4.5. Μορφές των ιχνών καταλληλότητας

Υπάρχουν δύο είδη ιχνών καταλληλότητας: τα ίχνη συσσώρευσης *(accumulating traces)*, στα οποία αναφερθήκαμε λίγο κατά την ανάλυση της προς τα πίσω προσέγγισης του TD(λ), και τα ίχνη αντικατάστασης *(replacing traces)* (Σχήμα 4.5).



Σχήμα 4.5 Τα accumulating (α) και replacing traces (β) για την κατάσταση *s* σε διάφορες χρονικές στιγμές. *(Το σχήμα προέρχεται από τους [Singh & Sutton 1994], Παράγραφος 1, Σχήμα1)*

Στα ίχνη συσσώρευσης *(accumulating traces)* το ίχνος καταλληλότητας μεγαλώνει κάθε φορά που μπαίνουμε σε μία νέα κατάσταση (Σχήμα 4.5 α). Μεγαλύτερη πίστωση δίνεται στις πιο πρόσφατες και στις πιο συχνά εμφανιζόμενες καταστάσεις (Sutton 1984). Ο τύπος βάσει του οποίου υπολογίζονται τα ίχνη

$$e_t(s) = \begin{cases} \gamma*\lambda*e_{t-1}(s)+1, & \text{αν } s=s_t \\ \gamma*\lambda*e_{t-1}(s), & \text{διαφορετικά} \end{cases}$$

στην περίπτωση αυτή είναι ο ακόλουθος:

όπου

$e_t(s)$*:* το ίχνος για την κατάσταση *s* τη χρονική στιγμή *t*.

λ: ο παράγοντας παράληψης *(forgetting factor)* με τιμές $0 \leq \lambda \leq 1$ *σ*

γ: η παράμετρος του ρυθμού μείωσης, μια σταθερά με τιμές $0 \leq \gamma < 1$.

Στα ίχνη αντικατάστασης *(replacing traces)* το ίχνος καταλληλότητας αρχικοποιείται με *1* κάθε φορά που μπαίνουμε σε μία νέα κατάσταση χωρίς να λαμβάνονται υπόψη τυχόν προηγούμενες τιμές. Το νέο ίχνος καταλληλότητας αντικαθιστά το παλιό (Σχήμα 4.5 β). Τα ίχνη αντικατάστασης δίνουν μεγαλύτερη πίστωση στις πιο πρόσφατες καταστάσεις γεγονός που μπορεί να επιφέρει σημαντική βελτίωση [Singh & Sutton 1994]. Ο τύπος βάσει του οποίου υπολογίζονται τα ίχνη καταλληλότητας στην περίπτωση αυτή είναι ο ακόλουθος:

$$e_t(s) = \begin{cases} 1, \text{ αν } s = s_t \\ \gamma * \lambda * e_{t-1}(s), \text{ διαφορετικά} \end{cases}$$

Ας υποθέσουμε ότι έχουμε επισκεφθεί μία κατάσταση και την ξανά επισκεπτόμαστε αμέσως μετά πριν το ίχνος από την προηγούμενη επίσκεψη προλάβει να εξασθενήσει στο μηδέν. Αν χρησιμοποιούσαμε ίχνη συσσώρευσης το ίχνος θα μεγάλωνε λόγω της αύξησης στη συχνότητα επίσκεψης και θα ξεπερνούσε το *1*, αντιθέτως αν χρησιμοποιούσαμε ίχνη αντικατάστασης το ίχνος θα είχε αρχικοποιηθεί στο *1* (Σχήμα 4.5).

Παρόλο που τα ίχνη αντικατάστασης δεν διαφέρουν πολύ από τα ίχνη συσσώρευσης η βελτίωση που επιφέρουν στο ρυθμό μάθησης είναι εξαιρετικά σημαντική. Οι μέθοδοι που χρησιμοποιούν ίχνη αντικατάστασης καλούνται μέθοδοι αντικατάστασης ίχνους *(replace - trace methods)*.

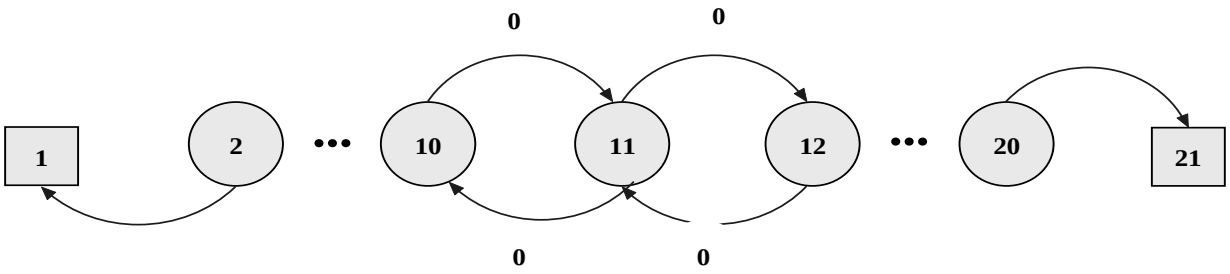
Στη συνέχεια παραθέτουμε ένα πείραμα, [Singh & Sutton 1995] - Παράγραφος 4, προκειμένου να συγκρίνουμε τα ίχνη αντικατάστασης με τα ίχνη συσσώρευσης.

4.6. Το πείραμα του τυχαίου περιπάτου (random walk experiment)

Ας θεωρήσουμε το παράδειγμα του τυχαίου περιπάτου [Singh & Sutton 1995] (Σχήμα 4.6).

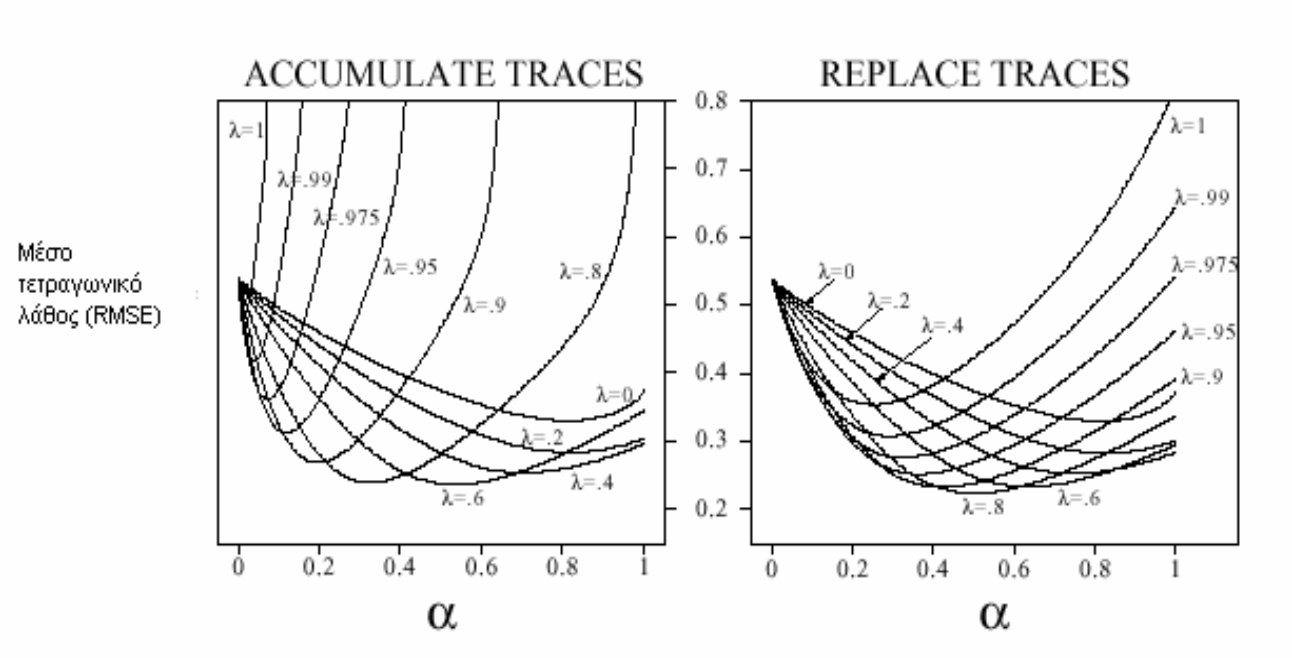
Ξεκινώντας από την κατάσταση *11* μπορούμε να προχωράμε αριστερά ή δεξιά με την ίδια πιθανότητα μέχρι να φτάσουμε στην κατάσταση *1* ή *21* (τελικές καταστάσεις). Η πίστωση είναι παντού 0 εκτός και αν πρόκειται για κάποια τελική κατάσταση. Στην τελευταία περίπτωση, αν η επόμενη κατάσταση είναι η *21* η πίστωση είναι *+1* ενώ αν η επόμενη κατάσταση είναι η *1* η πίστωση είναι *–1*. Η παράμετρος εξασθένισης του ίχνους *γ* είναι *1*.

Το πείραμα συνίσταται στην εφαρμογή του on-line TD (λ) για *10* διαφορετικές τιμές του *λ*: *0.0, 0.2, 0.4, 0.6, 0.8, 0.9, 0.95, 0.975, σ0.99* και *1* και για τα δύο είδη ιχνών. Η βηματική παράμετρος *α* διατηρήθηκε σταθερή, $\alpha_t(s)=\alpha$. Για κάθε τιμή του *λ*, χρησιμοποιήθηκαν τιμές του *α* μεταξύ *0* και *1* αυξανόμενες κάθε φορά κατά *0.01*. Κάθε ζεύγος τιμών *(α,λ)* θεωρήθηκε ως ξεχωριστός αλγόριθμος και έτρεξε για *10* πειράματα.



Σχήμα 4.6 Το παράδειγμα του τυχαίου περιπάτου *(Προσαρμόσητκε από τους [Singh & Sutton 1995], Παράγραφος 4, Σχήμα5)*

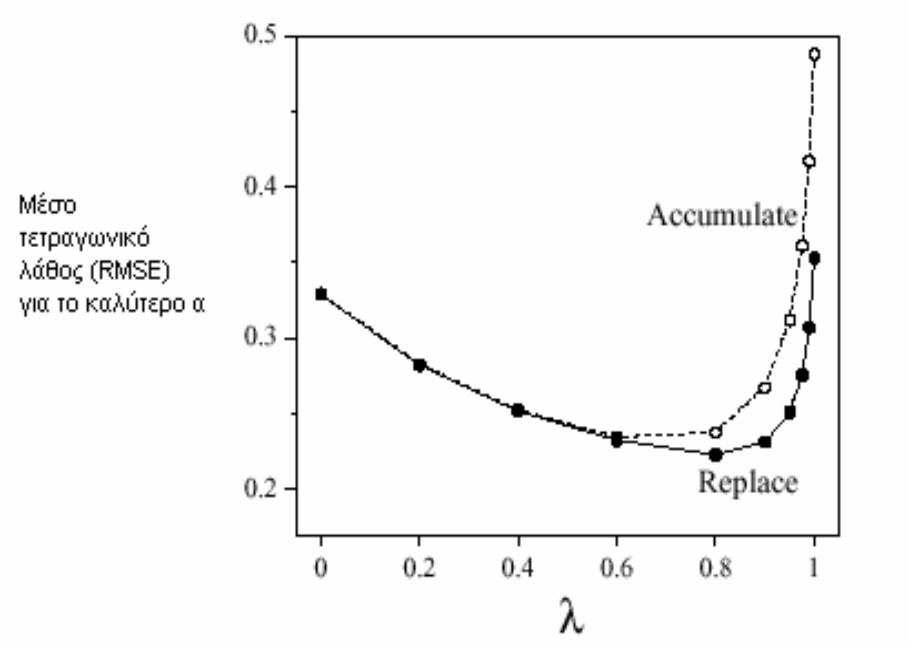
Για κάθε πείραμα μετρήθηκε η απόδοση που στην προκειμένη περίπτωση ήταν το μέσο τετραγωνικό λάθος *(root mean square error- RMSE)* μεταξύ των σωστών προβλέψεων και των προβλέψεων που έγιναν στο τέλος του πειράματος από τις καταστάσεις που τις είχαμε επισκεφθεί μία τουλάχιστον φορά κατά τη διάρκεια του τρέχοντος ή του προηγούμενου πειράματος. Οι αποδόσεις τόσο για τα ίχνη συσσώρευσης όσο και για τα ίχνη αντικατάστασης παριστάνονται γραφικά στο Σχήμα 4.7



Σχήμα 4.7 Η απόδοση των ιχνών αντικατάστασης και συσσώρευσης για τις διάφορες τιμές *λ,μ.(Το σχήμα προέρχεται από τους [Singh & Sutton 1995], Παράγραφος 4, Σχήμα 6)*

Όπως φαίνεται και στο παραπάνω σχήμα καθώς μεγαλώνει το *λ* θα πρέπει να μειωθεί το *α* για να επιτύχουμε καλή απόδοση, γεγονός που οφείλεται στο ότι στην εξίσωση ανανέωσης υπάρχει ο παράγοντας *λ*α*. Ωστόσο τα ίχνη αντικατάστασης αποδεικνύονται πιο σταθερά απέναντι στις αλλαγές του *α*. Πράγματι, για *λ≥0.9* τα ίχνη συσσώρευσης είναι ασταθή για τιμές του α*≥0.6.* Αυτό οφείλεται στο γεγονός ότι για μεγάλα *λ* τα ίχνη συσσώρευσης φτιάχνουν πολύ μεγάλα ίχνη καταλληλότητας για τις καταστάσεις που τις έχουν επισκεφθεί πολύ συχνά πριν τον τερματισμό με αποτέλεσμα οι υπολογιζόμενες τιμές να αλλάζουν πολύ και το όλο σύστημα να είναι ασταθές. Στο Σχήμα 4.8 παρουσιάζονται και πάλι οι δύο

μορφές ιχνών καταλληλότητας μόνο που τώρα παριστάνεται η απόδοση για κάθε *λ* και για εκείνα τα *α* που έδωσαν την καλύτερη απόδοση για το εκάστοτε *λ.*



Σχήμα 4.8 Οι καλύτερες αποδόσεις για τα ίχνη αντικατάστασης και συσσώρευσης στο πείραμα του τυχαίου περιπάτου.*(Προσαρμόστηκε από τους [Singh & Sutton 1995], Παράγραφος 4, Σχήμα 7)*

4.7. Θέματα υλοποίησης

Είναι εύλογο να αναρωτηθεί κανείς κατά πόσο τα ίχνη καταλληλότητας αυξάνουν την πολυπλοκότητα δεδομένου ότι σε κάθε βήμα θα πρέπει να ανανεώνουμε τόσο την αξία της εκάστοτε κατάστασης όσο και το ίχνος καταλληλότητας αυτής. Η απάντηση είναι πως χρησιμοποιώντας ίχνη καταλληλότητας η πολυπλοκότητα αυξάνεται, απ' την άλλη όμως η μάθηση είναι πιο γρήγορη και αποτελεσματική.

Εξάλλου, το πρόβλημα αυτό δεν είναι τόσο σημαντικό, καθώς στην πλειοψηφία των περιπτώσεων τα ίχνη καταλληλότητας είναι σχεδόν μηδέν για όλες τις καταστάσεις πλην αυτών που έχουμε επισκεφθεί πρόσφατα. Μόνο οι τελευταίες λίγες καταστάσεις χρειάζεται να ανανεωθούν, αφού τυχόν ανανέωση στις υπόλοιπες δε θα έχει ουσιαστικά αποτελέσματα.

Στην πράξη ανανεώνουμε μόνο τις λίγες πιο πρόσφατα επισκεπτόμενες καταστάσεις στις οποίες τα ίχνη καταλληλότητας είναι μη μηδενικά, με αποτέλεσμα η υπολογιστική πολυπλοκότητα να μειώνεται σημαντικά.

5. Νευρωνικά δίκτυα *(Neural networks)*

5.1. Εισαγωγή

Τα νευρωνικά δίκτυα αποτελούν έναν εύρωστο τρόπο προσέγγισης πραγματικών, διακριτών και διανυσματικών επιθυμητών συναρτήσεων *(target functions).* Έχουν εφαρμογή σε πολλές περιπτώσεις προβλημάτων όπως: η αναγνώριση προτύπων, η κατηγοριοποίηση, τα συστήματα ομιλίας, τα συστήματα όρασης, τα συστήματα ελέγχου κ.α. Μάλιστα για συγκεκριμένου είδους προβλήματα, όπως αυτό της μεταγλώττισης πολύπλοκων πραγματικών δεδομένων, τα νευρωνικά δίκτυα αποτελούν την πιο αποτελεσματική από τις υπάρχουσες μεθόδους αντιμετώπισης.

Η μελέτη των νευρωνικών υποκινήθηκε εν μέρει από την παρατήρηση ότι τα βιολογικά συστήματα μάθησης αποτελούνται από πολλούς διασυνδεόμενους νευρώνες. Ο ανθρώπινος εγκέφαλος για παράδειγμα υπολογίζεται ότι περιέχει ένα πυκνά διασυνδεόμενο σύνολο από περίπου 10^{11} νευρώνες, ο καθένας από τους οποίους συνδέεται με περίπου 10^4 νευρώνες. Έχει τη δυνατότητα να οργανώνει τους νευρώνες έτσι ώστε να εκτελεί συγκεκριμένους υπολογισμούς πολύ πιο γρήγορα από τους γρήγορους ψηφιακούς υπολογιστές που υπάρχουν.

Ένα νευρωνικό δίκτυο είναι ένας συμπαγής παράλληλος κατανεμημένος επεξεργαστής που έχει τη φυσική κλίση να αποθηκεύει εμπεριστατωμένη γνώση και να την κάνει διαθέσιμη για χρήση. Μοιάζει με τον εγκέφαλο στο ότι η γνώση αποκτάται από το δίκτυο μέσω μιας διαδικασίας μάθησης και οι δυνάμεις σύνδεσης των νευρώνων, γνωστές ως συναπτικά *(synaptic)* βάρη χρησιμοποιούνται για την αποθήκευση της γνώσης.

5.2. Ορισμός της έννοιας της μάθησης

Η πιο σημαντική ιδιότητα ενός νευρωνικού δικτύου είναι η ικανότητά του να μαθαίνει απ' το περιβάλλον του και έτσι να βελτιώνει την απόδοσή του μέσω της μάθησης. Η βελτίωση γίνεται σταδιακά με το χρόνο, σύμφωνα με κάποιο προκαθορισμένο μέτρο γνωστό ως ο ρυθμός μάθησης *(learning rate).* Η μάθηση επιτυγχάνεται μέσω μιας επαναλαμβανόμενης διαδικασίας ανανέωσης των βαρών των συνδέσεων του δικτύου. Θεωρητικά, το δίκτυο αποκτά περισσότερη γνώση για το περιβάλλον του μετά από κάθε επανάληψη της διαδικασίας μάθησης.

Βάσει του ορισμού των Mendel και McClaren η μάθηση στα νευρωνικά δίκτυα ορίζεται ως εξής:

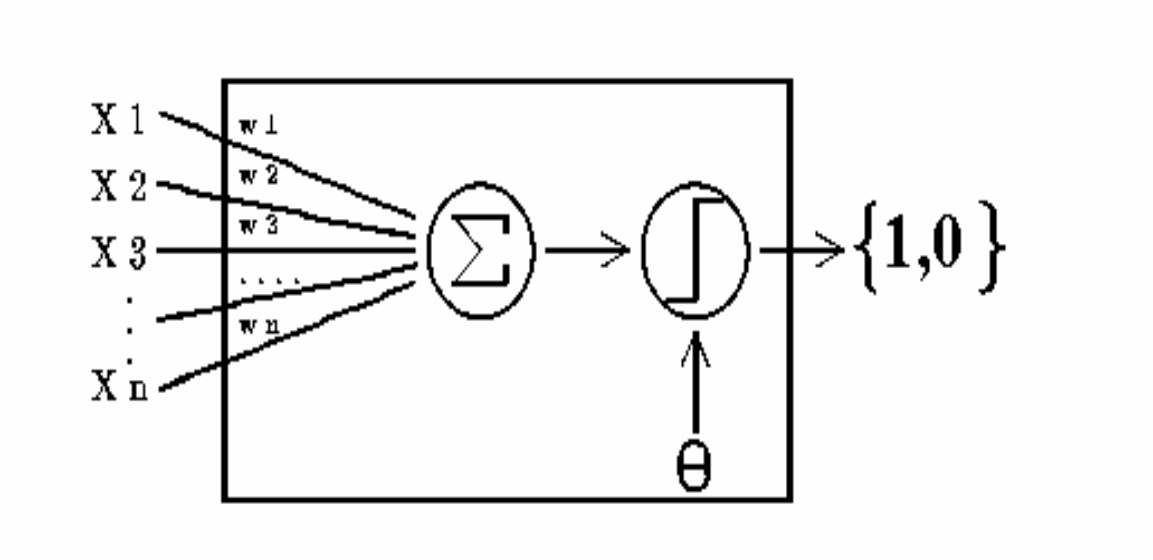
> Μάθηση είναι μια διαδικασία με την οποία προσαρμόζονται οι ελεύθεροι παράμετροι ενός νευρωνικού δικτύου μέσω μιας συνεχούς διαδικασίας διέγερσης από το περιβάλλον στο οποίο βρίσκεται το δίκτυο. Το είδος της μάθησης καθορίζεται από τον τρόπο με τον οποίο πραγματοποιούνται οι αλλαγές των παραμέτρων.

Ο παραπάνω ορισμός αφήνει να εννοηθεί ότι η διαδικασία μάθησης αποτελείται από τα ακόλουθα βήματα:

1. Το νευρωνικό δίκτυο «διεγείρεται» από ένα περιβάλλον
2. Το νευρωνικό δίκτυο υφίσταται αλλαγές σαν συνέπεια αυτής της διέγερσης
3. Το νευρωνικό δίκτυο απαντά με ένα καινούριο τρόπο στο περιβάλλον λόγω των αλλαγών που συνέβησαν στην εσωτερική του δομή.

5.3. Το perceptron

Η πιο απλή μορφή ενός νευρωνικού δικτύου είναι το perceptron (Σχήμα 5.1) το οποίο παίρνει ως είσοδο ένα διάνυσμα πραγματικών τιμών, υπολογίζει ένα γραμμικό συνδυασμό των εισόδων και δίνει ως έξοδο *1* αν το αποτέλεσμα είναι μεγαλύτερο από κάποιο κατώφλι *θ* ή *0* διαφορετικά.



Σχήμα 5.1 Το perceptron

Πιο συγκεκριμένα, δοθέντων των εισόδων $x_1, x_2,\ldots,x_n$ και του κατωφλίου θ, η έξοδος $o(x_1, x_2,\ldots,x_n)$ υπολογίζεται από το perceptron βάσει του τύπου:

$$o(x_1,x_2,\ldots,x_n) = \begin{cases} 1, \text{ αν } w_1*x_1+\ldots+w_n*x_n>\theta \\ 0, \text{αλλιώς} \end{cases}$$

Τα w_i είναι τα βάρη των συνδέσεων. Πρόκειται για σταθερές που αντικατοπτρίζουν κατά πόσο η είσοδος x_i επηρεάζει την έξοδο του perceptron. Σκοπός του perceptron είναι να ταξινομήσει τις εισόδους σε μία από τις κλάσεις εξόδου.

Η εκμάθηση του perceptron συνίσταται στον υπολογισμό των βαρών w_i. Κάθε διάνυσμα εισόδου παρουσιάζεται επαναληπτικά στο perceptron μέχρι να ταξινομηθεί σωστά. Το θεώρημα σύγκλισης του perceptron λέει πως η διαδικασία μάθησης συγκλίνει αν το σύνολο των παραδειγμάτων που χρησιμοποιούνται για την εκπαίδευση του δικτύου είναι γραμμικά διαχωρίσιμο (Σχήμα 5.2 α).

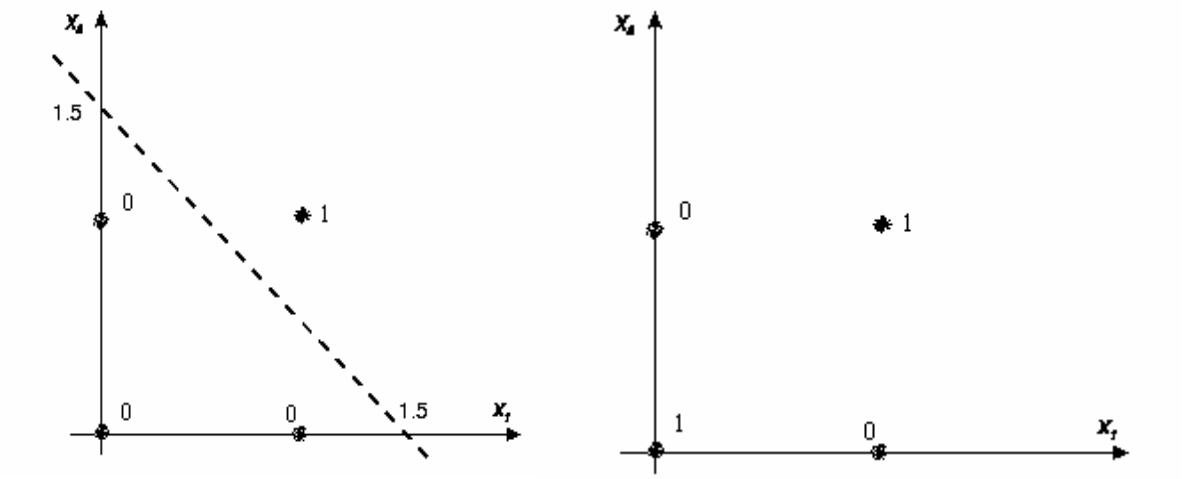
5.4. Ο χώρος των παραδειγμάτων εισόδου (input space)

Ας θεωρήσουμε ένα απλό νευρωνικό με εισόδους x1, x2 και βάρη w1, w2 αντίστοιχα. Έστω επίσης θ=1,5. Στον ακόλουθο πίνακα φαίνεται η ανταπόκριση του νευρωνικού ανάλογα με κάθε είσοδο.

X_1	x_2	Ενεργοποίηση εισόδου	Έξοδος
0	0	0	0
0	1	1	0
1	0	1	0
1	1	2	1

Πίνακας 5.1 Παράδειγμα ανταπόκρισης του νευρωνικού

Βάσει και του πίνακα το νευρωνικό μπορεί να θεωρηθεί ως η προσπάθεια κατηγοριοποίησης των εισόδων σε δύο κλάσεις: αυτές με έξοδο 1 και αυτές με έξοδο 0. Ας αναπαραστήσουμε γραφικά τις εισόδους σε ένα διδιάστατο χώρο (Σχήμα 5.2 α):



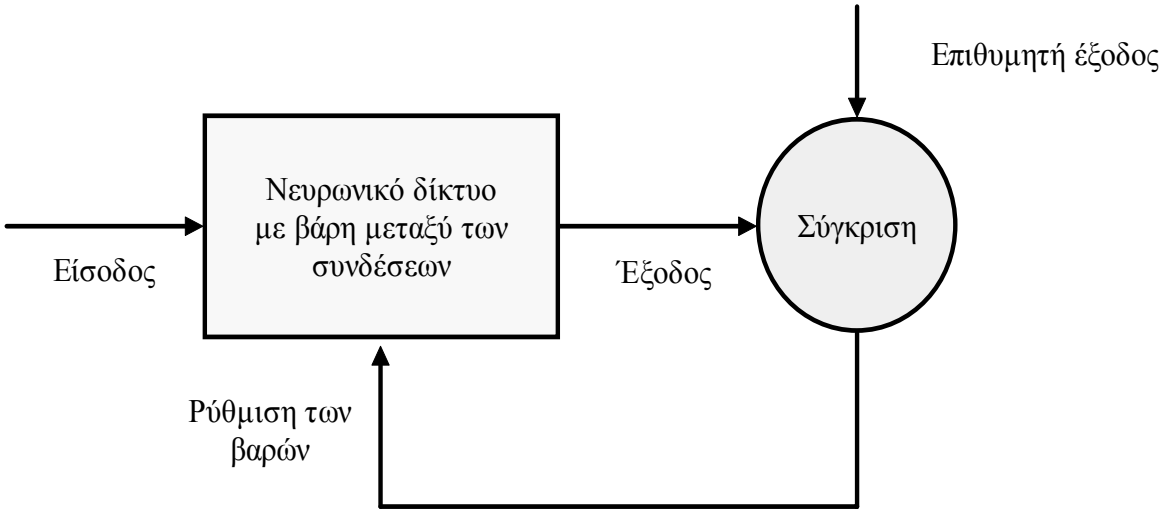
Σχήμα 5.2 (α) Γραμμικά διαχωρίσιμο σύνολο εκπαιδευτικών παραδειγμάτων **(β)** Μη γραμμικά διαχωρίσιμο σύνολο εκπαιδευτικών παραδειγμάτων

Στο παραπάνω σχήμα (Σχήμα 5.2 α) φαίνεται πως οι δύο κλάσεις είναι διαχωρίσιμες με μια ευθεία, στην περίπτωση αυτή λέμε ότι το σύνολο των εκπαιδευτικών παραδειγμάτων εισόδου είναι γραμμικά διαχωρίσιμο και το perceptron μπορεί να τα ταξινομήσει σωστά. Η ευθεία που διαχωρίζει τις δύο κλάσεις ονομάζεται όριο απόφασης *(decision boundary)* και η μορφή της καθορίζεται από το *θ.* Στην αντίθετη περίπτωση, όταν δηλαδή τα εκπαιδευτικά παραδείγματα εισόδου δεν είναι γραμμικά διαχωρίσιμα, (Σχήμα 5.2 β) η ταξινόμηση είναι πιο δύσκολη και δεν μπορεί να γίνει με χρήση ενός απλού perceptron. Στην περίπτωση αυτή, χρειάζονται πιο πολύπλοκες δομές νευρωνικών δικτύων που ονομάζονται perceptrons πολλών επιπέδων.

5.5. Κατανοώντας τα νευρωνικά δίκτυα

Τα νευρωνικά αποτελούνται από επιμέρους μονάδες που λειτουργούν παράλληλα. Η συνάρτηση του δικτύου καθορίζεται ως επί το πλείστον από τις συνδέσεις μεταξύ των perceptrons. Μπορούμε να εκπαιδεύσουμε το νευρωνικό ώστε να εκτελεί μία συγκεκριμένη συνάρτηση ρυθμίζοντας τα βάρη μεταξύ των συνδέσεων.

Συνήθως τα νευρωνικά εκπαιδεύονται ώστε μία συγκεκριμένη είσοδος να οδηγεί σε μία συγκεκριμένη έξοδο, όπως φαίνεται στο Σχήμα 5.3. Στη συνέχεια το νευρωνικό ρυθμίζεται βάσει μιας σύγκρισης της τρέχουσας εξόδου με την επιθυμητή έξοδο, μέχρι να ταιριάξουν.



Σχήμα 5.3 Λειτουργία ενός νευρωνικού

Η μάθηση επιτυγχάνεται εφαρμόζοντας ένα σύνολο από διανύσματα εκπαίδευσης ως είσοδο στο νευρωνικό. Η προβολή όλων των διανυσμάτων εκπαίδευσης στο νευρωνικό λέγεται κύκλος *(epoch).* Η διαδικασία μάθησης προχωράει από epoch σε epoch, μέχρι να σταθεροποιηθούν τα βάρη του δικτύου και το λάθος εξόδου να τείνει σε κάποια ελάχιστη τιμή. Μια καλή πρακτική είναι να θέτουμε τα διανύσματα εκπαίδευσης σε μια τυχαία σειρά από μία epoch σε μία άλλη, έτσι αποφεύγουμε τον κίνδυνο να γίνουν λιγότεροι κύκλοι από ότι πρέπει.

Υπάρχουν δύο είδη εκπαίδευσης: η εκπαίδευση παρτίδας *(batch training)* και η αυξητική εκπαίδευση *(incremental training).*

Στην εκπαίδευση παρτίδας η ανανέωση των βαρών πραγματοποιείται στο τέλος της εμφάνισης όλων των διανυσμάτων εκπαίδευσης που αποτελούν ένα epoch. Η ανανέωση των βαρών γίνεται βάσει της σχέσης:

$$\Delta w_i = -\eta \frac{\delta E_{av}}{\delta w}$$

όπου E_{av}: το μέσο τετραγωνικό λάθος που δίνεται από τη σχέση:

$$E_{av} = \frac{1}{2N} \sum_{n=1}^{N} \sum_{j \in C} e_j^2(n)$$

Στην αυξητική εκπαίδευση, απ' την άλλη η ανανέωση των βαρών πραγματοποιείται μετά την εμφάνιση κάθε μεμονωμένου εκπαιδευτικού διανύσματος. Η μέση αλλαγή των βαρών του δικτύου για όλα τα διανύσματα (πλήθος N π.χ.) ενός epoch δίνεται από τη σχέση:

$$\Delta w = \frac{1}{N} \sum_{n=1}^{N} \Delta w(n) = -\frac{\eta}{N} \sum_{n=1}^{N} e(n) \frac{\delta e(n)}{\delta w(n)}$$

Το πλεονέκτημα της αυξητικής εκπαίδευσης είναι ότι χρειάζεται λιγότερη μνήμη για κάθε σύνδεση και δεδομένου ότι τα διανύσματα προβάλλονται στο δίκτυο με τυχαία σειρά το ψάξιμο στο χώρο των βαρών μετατρέπεται σε στοχαστική διαδικασία γλιτώνοντας έτσι τον αλγόριθμο από την παγίδα των τοπικών ελαχίστων. Η εκπαίδευση παρτίδας από την άλλη δίνει πιο σωστή εκτίμηση του gradient vector. Καταλήγουμε λοιπόν, στο ότι η φύση του προβλήματος που θέλουμε να αντιμετωπίσουμε καθορίζει και το είδος της εκπαίδευσης που πρέπει να χρησιμοποιήσουμε.

5.6. Το πρόβλημα της Ανάθεσης Πίστωσης *(Credit Assignment)*

Το πρόβλημα της Ανάθεσης Πίστωσης ορίζεται ως το πρόβλημα της απόδοσης επαίνου *(reward)* ή μομφής *(blame)* για τα συνολικά αποτελέσματα σε κάθε μια εσωτερική απόφαση που πήρε το σύστημα και είχε επίδραση σ' αυτά τα αποτελέσματα. Πολλές φορές οι εσωτερικές αποφάσεις καθορίζουν ποιες ενέργειες γίνονται και στη συνέχεια οι ενέργειες αυτές επηρεάζουν άμεσα το τελικό αποτέλεσμα. Σ' αυτές τις περιπτώσεις μπορούμε να διασπάσουμε το πρόβλημα της ανάθεσης πίστωσης σε δύο υποπροβλήματα:

1. Ανάθεση πίστωσης για τα αποτελέσματα στις ενέργειες *(temporal credit assignment):* το πρόβλημα αυτό σχετίζεται με τις περιπτώσεις όπου γίνονται πολλές ενέργειες από το σύστημα μάθησης και οδηγούν σε συγκεκριμένα αποτελέσματα.. Θα πρέπει να αποφασιστεί ποιες από τις ενέργειες αυτές ήταν στην πραγματικότητα υπεύθυνες για τα αποτελέσματα. ώστε οι ενέργειες αυτές να ανταμειφθούν ανάλογα.
2. Ανάθεση πίστωσης για τις ενέργειες στις εσωτερικές αποφάσεις *(structural credit assignment):* το πρόβλημα αυτό σχετίζεται με συστήματα μάθησης πολλών στοιχείων, όπου υπάρχει το πρόβλημα να αποφασιστεί ποιο συγκεκριμένο στοιχείο πρέπει να μεταβάλλει τη συμπεριφορά του και πόσο ώστε να βελτιωθεί η συνολική απόδοση του συστήματος.

Υπάρχουν αρκετοί αλγόριθμοι μάθησης που διαφέρουν ως προς τον τρόπο με τον οποίο υπολογίζουν την ανανέωση των βαρών. Από τους πιο δημοφιλείς κανόνες ανανέωσης των βαρών είναι ο κανόνας του perceptron *(*perceptron *rule)* και ο κανόνας του δέλτα *(delta rule)*.

5.7. Ο κανόνας του perceptron (Perceptron rule)

Η βασική ιδέα είναι η εξής: ξεκινάμε με αυθαίρετα βάρη και στη συνέχεια ενεργοποιούμε το perceptron εφαρμόζοντας τα διανύσματα εκπαίδευσης. Στην περίπτωση που κάποιο από τα παραδείγματα αυτά δεν ταξινομείται σωστά, αλλάζουμε τα βάρη του δικτύου. Η παραπάνω διαδικασία επαναλαμβάνεται ως ότου το perceptron ταξινομήσει σωστά όλα τα παραδείγματα εισόδου. Τα βάρη ανανεώνονται σε κάθε βήμα βάσει της σχέσης:

$$w_i \leftarrow w_i + \eta\,(t - o)\,x_i$$

όπου

t: η επιθυμητή έξοδος *(desired response).* Είναι *t=1*, αν η είσοδος ανήκει στην κλάση *1* και *t=0* διαφορετικά.

o: η πραγματική έξοδος *(actual response)*

x: το διάνυσμα εισόδου

η: ο ρυθμός μάθησης *(learning rate).* Πρόκειται για μια θετική σταθερά με μικρή τιμή *(<1)*, Ο ρόλος της είναι να αλλάζει το βαθμό στον οποίο τα βάρη ανανεώνονται σε κάθε βήμα.

Ας δούμε τώρα αν ο κανόνας αυτός συγκλίνει στα σωστά βάρη εξετάζοντας κάποιες ακραίες περιπτώσεις.

- Αν το νευρωνικό ταξινομεί σωστά το παράδειγμα εισόδου, τότε τα βάρη του δικτύου θα παραμείνουν αμετάβλητα.
- Αν η έξοδος του νευρωνικού είναι –1, ενώ η επιθυμητή έξοδος είναι +1, τότε τα βάρη θα πρέπει να αυξηθούν. Πράγματι η παράσταση $\eta(t-o)x_i$ είναι θετική, συνεπώς η νέα τιμή του w_i είναι μεγαλύτερη.

- Στην αντίθετη περίπτωση, αν η έξοδος του νευρωνικού είναι +1, ενώ η επιθυμητή έξοδος είναι -1, τότε τα βάρη θα πρέπει να μειωθούν. Πράγματι η παράσταση $\eta(t-o)x_i$ είναι αρνητική, συνεπώς η νέα τιμή του w_i είναι μικρότερη.

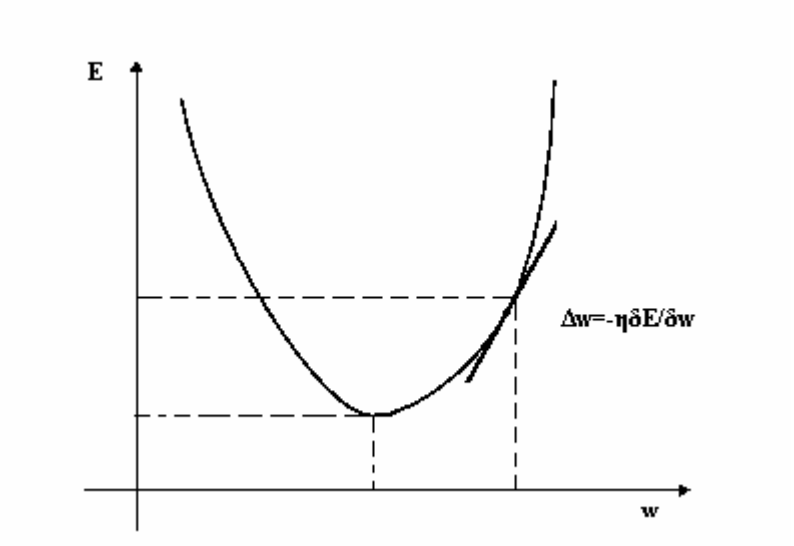
Ο κανόνας του perceptron αποδεικνύεται ότι συγκλίνει πάντα όταν το σύνολο των εκπαιδευτικών παραδειγμάτων εισόδου είναι γραμμικά διαχωρίσιμο και ο ρυθμός μάθησης είναι αρκετά μικρός.

5.8. Ο κανόνας του δέλτα (Delta rule)

Είδαμε ότι ο κανόνας του perceptron συγκλίνει με την προϋπόθεση ότι το σύνολο των εκπαιδευτικών παραδειγμάτων εισόδου είναι γραμμικά διαχωρίσιμο. Για την περίπτωση όμως που το σύνολο των εκπαιδευτικών παραδειγμάτων εισόδου δεν είναι γραμμικά διαχωρίσιμο χρησιμοποιείται ο κανόνας του δέλτα, ο οποίος συγκλίνει στην πιο καλή προσέγγιση της επιθυμητής εξόδου.

Η βασική ιδέα στηρίζεται στην ελαχιστοποίηση του λάθους στους νευρώνες εξόδου. Ψάχνουμε δηλαδή, εκείνα τα βάρη w_{ij} για τα οποία η διαφορά της πραγματικής εξόδου από την επιθυμητή έξοδο του δικτύου είναι η ελάχιστη δυνατή.

Προκειμένου να βρούμε στο χώρο των πιθανών λύσεων τη λύση εκείνη που ελαχιστοποιεί το λάθος στους νευρώνες εξόδου χρησιμοποιούμε τη μέθοδο ταχύτερης καθόδου *(steepest descent method)*, η οποία έχει σαν στόχο τη συνεχή αναζήτηση βέλτιστης λύσης. Όπως φαίνεται και στο Σχήμα 5.4, οι διορθώσεις για να είναι επιτυχημένες θα πρέπει να γίνονται σε κατεύθυνση αντίθετη του διανύσματος τελεστή (gradient vector) *δE/Δw.*



Σχήμα 5.4 Το διάνυσμα τελεστής *(gradient vector) δE/Δw*

Το διάνυσμα τελεστής *(gradient vector)* είναι ένας παράγοντας ευαισθησίας *(sensitivity factor)* που καθορίζει την κατεύθυνση έρευνας στο χώρο των βαρών *(weight space)* για το συναπτικό βάρος w.

Τα βάρη ανανεώνονται βάσει της σχέσης:

$$w_i = w_i - \eta \frac{\partial E}{\partial w_i}$$

το μείον ερμηνεύεται σαν πτώση του διανύσματος τελεστή στο χώρο των βαρών. Το *E είναι* το τετραγωνικό λάθος της εξόδου και δίνεται από τη σχέση:

Συνοψίζοντας, ο κανόνας delta δουλεύει ως εξής: Επιλέγουμε ένα τυχαίο διάνυσμα βαρών και εφαρμόζουμε όλα τα εκπαιδευτικά παραδείγματα στο νευρωνικό. Υπολογίζουμε το *Δw* και ανανεώνουμε

$$E \equiv \frac{1}{2}\sum_{d \in D}(t_d - o_d)^2$$

τις τιμές των βαρών. Επαναλαμβάνουμε την παραπάνω διαδικασία για όλα τα παραδείγματα εισόδου. Ο αλγόριθμος τελικά συγκλίνει στο διάνυσμα βαρών με το μικρότερο λάθος, ανεξάρτητα από το αν τα παραδείγματα που χρησιμοποιούνται για την εκπαίδευση του δικτύου είναι γραμμικά διαχωρίσιμα ή όχι, με την προϋπόθεση βέβαια ότι ο ρυθμός μάθησης είναι μικρός. Αν το η είναι πολύ μεγάλο, υπάρχει ο κίνδυνος να υπερβούμε το ελάχιστο και για το λόγο αυτό συχνά μειώνουμε την τιμή του *η* καθώς το πλήθος των βημάτων για την αναζήτηση του ελαχίστου αυξάνεται.

5.9. Ο ρυθμός μάθησης *(learning rate)*

Ας εξετάσουμε πιο λεπτομερειακά την επίδραση του ρυθμού μάθησης *λ*. Όσο πιο μικρό είναι το *λ,* τόσο πιο μικρές θα είναι σε κάθε επανάληψη του συνόλου των εκπαιδευτικών παραδειγμάτων οι αλλαγές στα βάρη του δικτύου και τόσο πιο ομαλή θα είναι η σύγκλιση. Αυτό όμως έχει σαν αντίτιμο πιο αργό ρυθμό μάθησης. Από την άλλη, αν αυξήσουμε το *λ* προκειμένου να επιταχύνουμε το ρυθμό μάθησης, οι μεγάλες αλλαγές που θα υπάρξουν στα βάρη μπορεί να οδηγήσουν σε ένα ασταθές δίκτυο.

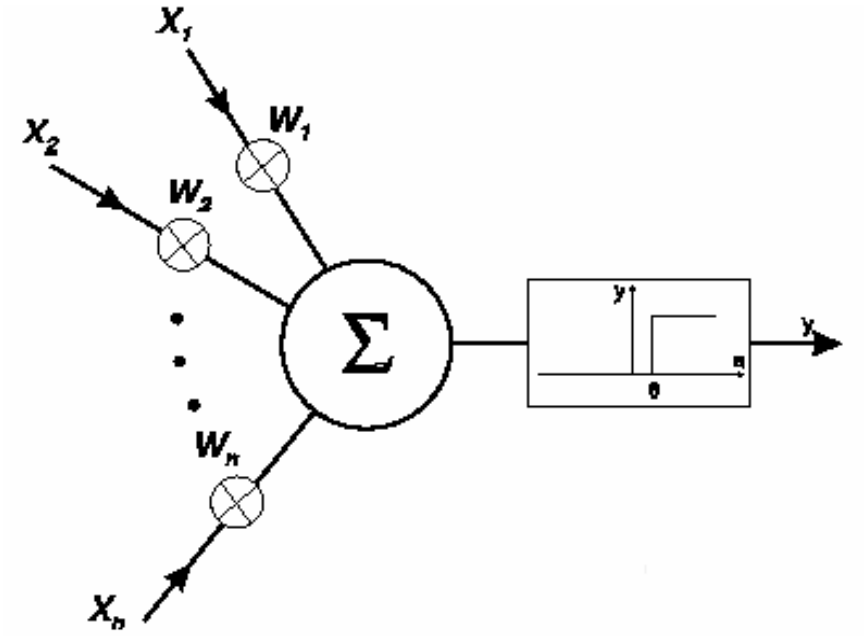
Μια απλή μέθοδος να αυξήσουμε το *λ* και ταυτόχρονα να αποφύγουμε τον κίνδυνο της αστάθειας είναι να χρησιμοποιήσουμε κατά την ανανέωση των βαρών του δικτύου και έναν όρο ορμής (momentum term) *α.* Πρόκειται συνήθως για έναν θετικό αριθμό που καθορίζει πόσο μεγάλη είναι η αλλαγή του βάρους στον επόμενο υπολογισμό. Η ανανέωση των βαρών γίνεται πλέον ως εξής:

$$\Delta w(n) = a * \Delta w(n-1) + \eta t(n)(o(n)_i$$

Η συμπερίληψη του momentum στον αλγόριθμο backpropagation αποτελεί μια μικρή αλλαγή όσον αφορά την τροποποίηση των βαρών, αλλά έχει πολλές θετικές επιδράσεις στη μάθηση του αλγορίθμου και μπορεί να εμποδίσει τον τερματισμό της διαδικασίας σε ένα τοπικό ελάχιστο.

5.10. Συνάρτηση κατωφλίου

Η επιρροή που προκαλεί κάθε είσοδος στην έξοδο του νευρωνικού, υπολογίζεται πολλαπλασιάζοντας την είσοδο με το βάρος της σύνδεσης. Τα επιμέρους γινόμενα αθροίζονται παράγοντας τη συνολική ενεργοποίηση της εισόδου *(unit activation).* Εξετάζοντας την ενεργοποίηση αυτή υπολογίζουμε την ανταπόκριση του νευρωνικού *(output response)*. Έστω έχουμε ένα απλό perceptron με *n* εισόδους x_1, x_2, …x_n με τιμές 0 και 1 και βάρη w_1, w_2, …w_n. (Σχήμα 5.5)



Σχήμα 5.5 Παράδειγμα νευρωνικού

Η έξοδος του δίνεται από τη σχέση:

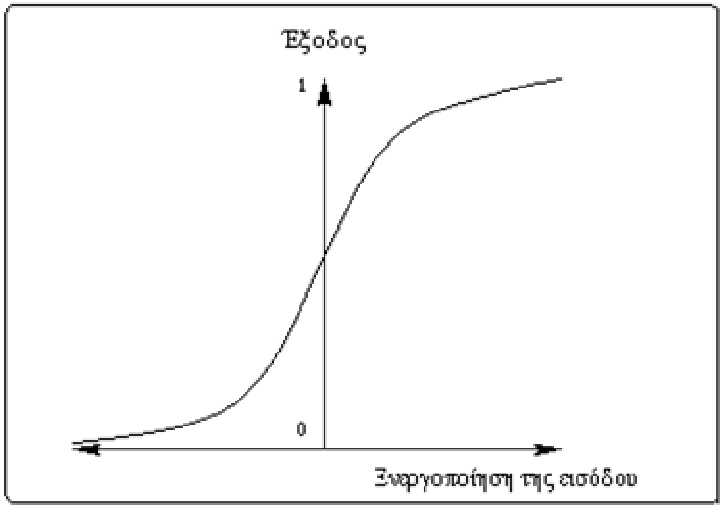
$$o(x_1,x_2,\ldots,x_n) = \begin{cases} 1, \text{ αν } w_1*x_1+\ldots+w_n*x_n>\theta \\ 0, \text{αλλιώς} \end{cases}$$

όπου θ είναι το κατώφλι, γνωστό ως bias (σε αρκετές περιπτώσεις ισούται με μηδέν). Η συνάρτηση κατωφλίου ονομάζεται συχνά και βηματική συνάρτηση *(step function)* ή ψαλιδιστής *(hard limiter)* –για προφανείς λόγους.

Στην περίπτωση πραγματικών πολύπλοκων νευρώνων και όχι του απλού perceptron που αναφέραμε μόλις πριν αντί να υπολογίσουμε μια δυαδική έξοδο, η ενεργοποίηση της εισόδου συναθροίζεται με το κατώφλι *θ* και το αποτέλεσμα περνάει μέσα από μία σιγμοειδή συνάρτηση (**Σχήμα 5.6)** της μορφής:

*y=ó(w1*x1+...+wn*xn)=1/(1+ e-(w1*x1+...+wn*xn))*

Η έξοδος της σιγμοειδούς είναι ένας πραγματικός αριθμός μεταξύ 0 και 1 και όχι δυαδική όπως στην περίπτωση του απλού perceptron. Η σιγμοειδής συνάρτηση καλείται και συνάρτηση πολτοποίησης *(squashing function),* επειδή ακριβώς αντιστοιχεί την είσοδο σε ένα συγκεκριμένο εύρος τιμών. Με τον τρόπο αυτό επιτυγχάνουμε την αντιστοίχηση πολλών τιμών της εισόδου σε περιοριστικές τιμές της εξόδου.



Σχήμα 5.6 Η σιγμοειδής συνάρτηση

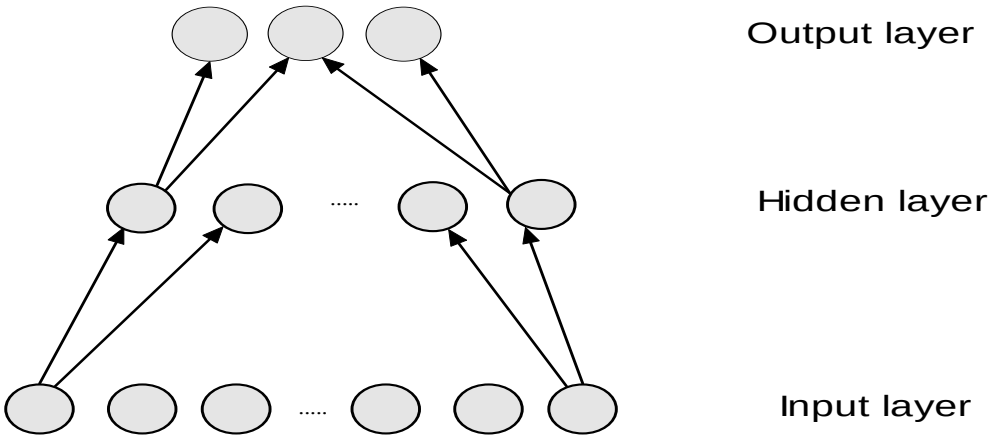
5.11. Perceptrons πολλών επιπέδων *(Multilayer perceptrons)*

Σε πολλά προβλήματα του πραγματικού κόσμου, αντιμετωπίζουμε το πρόβλημα των ελλιπών δεδομένων ή των δεδομένων που εμπεριέχουν θόρυβο, και γι' αυτό είναι σημαντικό να μπορούμε να κάνουμε λογικές προβλέψεις σχετικά με την πληροφορία που δεν έχουμε. Το έργο αυτό είναι πολύ δύσκολο καθώς δεν υπάρχει μία καλή θεωρία για την ανακατασκευή τη πληροφορίας. Σε αυτές τις περιπτώσεις η λύση είναι πιθανό να προέρχεται από τα νευρωνικά δίκτυα και πιο συγκεκριμένα από τα Perceptrons πολλών επιπέδων *(multi layer perceptrons-MLP),* τα οποία είναι μια γενίκευση του απλού perceptron.

Τα MLP είναι ανεκτικά στην παρουσία θορύβου: μικρές αλλαγές στην είσοδο ενός νευρώνα (λόγω θορύβου) δεν επηρεάζουν δραματικά την έξοδο του δικτύου.

Ένα τέτοιο δίκτυο αποτελείται από τρία τουλάχιστον επίπεδα:

το επίπεδο εισόδου (input layer), το κρυμμένο επίπεδο (hidden layer) και το επίπεδο εξόδου (output layer). Η συνήθης αρχιτεκτονική ενός τέτοιου δικτύου φαίνεται στο ακόλουθο σχήμα (Σχήμα 5.7).



Σχήμα 5.7 Η συνήθης αρχιτεκτονική ενός backpropagation δικτύου

Χαρακτηριστική είναι η παρουσία των κρυμμένων επιπέδων *(hidden layers)* και οι συνδέσεις αυτών με τα επίπεδα εισόδου και εξόδου. Οι συνδέσεις είναι εμπρόσθιες *(feed-forward),* δεν υπάρχουν δηλαδή συνδέσεις από τα κρυμμένα επίπεδα προς τα επίπεδα εισόδου, και χαρακτηρίζονται από κάποια βάρη.

Η όλη πληροφορία αποθηκεύεται στα βάρη των συνδέσεων , τα οποία προσαρμόζονται ανάλογα με την είσοδο του δικτύου.

Ο αλγόριθμος που χρησιμοποιείται στην πλειοψηφία των multiplayer perceptrons είναι ο αλγόριθμος της πίσω διάδοσης του λάθους *(error back propagation algorithm)*.

Η διαδικασία της πίσω διάδοσης του λάθους αποτελείται από δύο περάσματα διαμέσου των διαφορετικών επιπέδων του δικτύου: ένα πέρασμα προς τα εμπρός (forward pass) και ένα πέρασμα προς τα πίσω (backward pass).

- Στο εμπρός πέρασμα ένα διάνυσμα εισόδου *(input vector)* εφαρμόζεται στους νευρώνες εισόδου του δικτύου και η επίδρασή του διαδίδεται μέσα στο δίκτυο από επίπεδο σε επίπεδο *(layer by layer).*Ένα σύνολο από εξόδους παράγεται, αυτή είναι η πραγματική ανταπόκριση του δικτύου *(actual response of the network).* Κατά τη διάρκεια αυτού του περάσματος τα βάρη του δικτύου είναι σταθερά.
- Στο πέρασμα προς τα πίσω τα βάρη προσαρμόζονται βάσει του κανόνα διόρθωσης του λάθους *(error correction rule).* Πιο συγκεκριμένα, η πραγματική ανταπόκριση του δικτύου αφαιρείται από την επιθυμητή ανταπόκριση του δικτύου και υπολογίζεται το λάθος, το οποίο διαδίδεται προς τα πίσω στο δίκτυο αντίθετα από την κατεύθυνση των συνδέσεων (γι' αυτό ονομάστηκε και αλγόριθμος της πίσω διάδοσης του λάθους).Τα βάρη προσαρμόζονται έτσι ώστε η πραγματική ανταπόκριση του δικτύου να πλησιάσει την επιθυμητή απόδοση του δικτύου.

Επισημαίνουμε ότι κατά την παρουσίαση στο νευρωνικό οποιουδήποτε εκπαιδευτικού παραδείγματος, το διάνυσμα εισόδου παραμένει σταθερό *(fixed)* και αυτό ισχύει για όσο διαρκούν τα δύο περάσματα του αλγορίθμου.

Ένα από τα βασικά μειονεκτήματα των Multi layer perceptrons είναι η εξαιρετικά δύσκολη θεωρητική τους ανάλυση, γεγονός που οφείλεται στην υψηλή διασύνδεση του δικτύου και στην παρουσία μιας κατανεμημένης μορφής μη γραμμικότητας λόγω της σιγμοειδούς συνάρτησης. Ακόμη ένα πρόβλημα είναι

η παρουσία των κρυφών νευρώνων που δυσκολεύει τη διαδικασία μάθησης, γιατί κατά κάποιο τρόπο αυτή πρέπει να αποφασίσει για το ποια χαρακτηριστικά των διανυσμάτων εισόδου πρέπει να παρασταθούν από τους κρυφούς νευρώνες. Έτσι η έρευνα πρέπει να διεξάγεται σε ένα πολύ μεγαλύτερο χώρο από πιθανές συναρτήσεις και να γίνεται μια επιλογή μεταξύ εναλλακτικών αναπαραστάσεων του διανύσματος εισόδου.

5.12. Αλγόριθμος Backpropagation

Ο backpropagation αλγόριθμος αποτελεί σταθμό στα νευρωνικά δίκτυα καθώς παρέχει μια υπολογιστικά αποδοτική μέθοδο για την εκπαίδευση perceptrons πολλών επιπέδων. Ακόμη και μη γραμμικά διαχωρίσιμα προβλήματα, όπως αυτό της αναγνώρισης της φωνής, επιλύονται μέσω του παραπάνω αλγορίθμου. Περιγράψαμε στην προηγούμενη παράγραφο τη διαδικασία της πίσω διάδοσης του λάθους, ας δούμε όμως πιο αναλυτικά τι συμβαίνει. Όπως είπαμε και πριν, ο backpropagation βασίζεται στη μάθηση μέσω του λάθους, διορθώνει δηλαδή το σφάλμα στους νευρώνες εξόδου με στόχο την ελαχιστοποίησή του. Με τον όρο σφάλμα εννοούμε το άθροισμα των επιμέρους σφαλμάτων των νευρώνων εξόδου, δηλαδή το άθροισμα των αποκλίσεων των εξόδων του νευρωνικού από τις αντίστοιχες επιθυμητές εξόδους (*target output*). Δίνεται από τη σχέση:

$$E = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2$$

όπου

Outputs: το σύνολο των νευρώνων εξόδου του δικτύου

t_{kd}: η επιθυμητή έξοδος (desired output)

o_{kd}: η πραγματική έξοδος (actual output)

Ο Backpropagation ψάχνει στο χώρο των πιθανών λύσεων εκείνη τη λύση που ελαχιστοποιεί το λάθος στους νευρώνες εξόδου του δικτύου. Το μειονέκτημα είναι ότι μπορεί να υπάρχουν πολλά τοπικά ελάχιστα, με αποτέλεσμα η σύγκλισή του αλγορίθμου να γίνεται ως προς κάποιο τοπικό ελάχιστο και όχι ως προς το ολικό ελάχιστο. Ευτυχώς, αυτό δεν φαίνεται να επηρεάζει πολύ την απόδοση του, στην πράξη στην πλειοψηφία των περιπτώσεων τα αποτελέσματα είναι εξαιρετικά.

Αν και οι κρυφοί νευρώνες δεν είναι άμεσα προσπελάσιμοι ωστόσο μοιράζονται ευθύνη για κάθε λάθος που συμβαίνει στην έξοδο του δικτύου. Το ζήτημα είναι το πως θα επιβάλλουμε ποινή *(penalize)* ή θα επιβραβεύσουμε *(reward)* τους κρυφούς νευρώνες για το μερίδιο της ευθύνης τους. Το πρόβλημα αυτό είναι το Πρόβλημα της Ανάθεσης Πίστωσης *(Credit-Assignment Problem),* το οποίο αναφέραμε παραπάνω.

Μια μορφή του backpropagation αλγορίθμου που χρησιμοποιεί αυξητική μάθηση, δηλαδή η ανανέωση των βαρών του δικτύου γίνεται μετά την εμφάνιση κάθε μεμονωμένου διανύσματος εισόδου, παρουσιάζεται παρακάτω σε μορφή ψευδοκώδικα:

Αλγόριθμος backpropagation (εκπαιδευτικά παραδείγματα, η, n_{in},n_{out}, n_{hidden})

εκπαιδευτικά παραδείγματα: είναι διανύσματα της μορφής *(x,t)* όπου *x* είναι το διάνυσμα εισόδου του νευρωνικού και *t* οι αντίστοιχες επιθυμητές τιμές στην έξοδο *(target output)*.

η: ο ρυθμός μάθησης

n_{in}: το σύνολο των νευρώνων εισόδου

n_{hidden}: το σύνολο των νευρώνων εισόδου

n_{out}: το σύνολο των νευρώνων εξόδου

- Δημιούργησε ένα δίκτυο διασποράς προς τα πίσω με n_{in} εισόδους, n_{hidden} κρυμμένους νευρώνες και n_{out} εξόδους.
- Αρχικοποίησε τα βάρη του δικτύου με μικρές τιμές (π.χ. από -0.5 μέχρι 0.5)
- Μέχρι να ισχύει η συνθήκη τερματισμού κάνε τα ακόλουθα:

  Για κάθε (x,t) διάνυσμα στα εκπαιδευτικά παραδείγματα κάνε τα εξής:

  Διέδωσε την είσοδο προς τα μπρος μέσω του δικτύου

  1. Βάλε την είσοδο x στο δίκτυο και υπολόγισε την έξοδο o_u για κάθε νευρώνα u του δικτύου.

  Διέδωσε τα λάθη προς τα πίσω μέσω του δικτύου

  2. Για κάθε νευρώνα εξόδου k υπολόγισε το λάθος δ_k

  $$\delta_k \leftarrow o_k(1-o_k)(t_k - o_k)$$

  3. Για κάθε κρυφό νευρώνα k υπολόγισε το λάθος δ_h

  $$\delta_h \leftarrow o_h(1-o_h)\sum_{k \in outputs} w_{kh}\delta_k$$

  4. Ανανέωσε κάθε βάρος w_{ji} του δικτύου.

  $$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

6. Μοντελοποίηση παικτών

6.1. Εισαγωγή

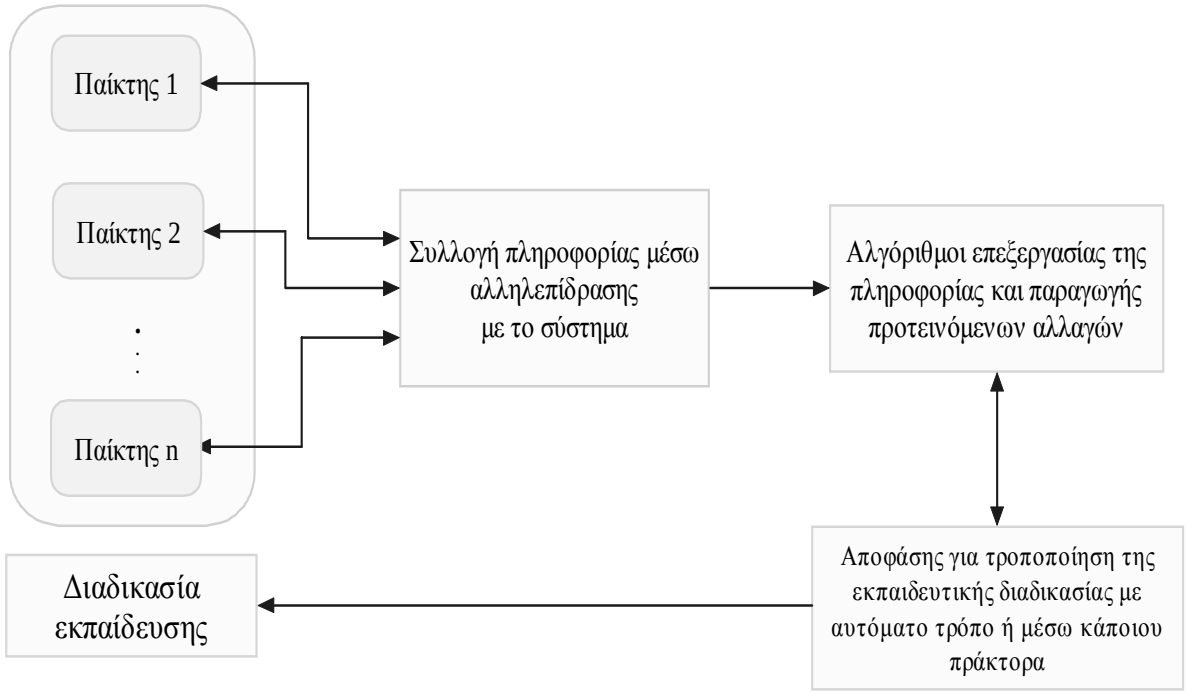
Με τον όρο μοντελοποίηση αναφερόμαστε στην αυτόματη συλλογή και επεξεργασία στοιχείων σχετικά με τη συμπεριφορά των παικτών και αποβλέπουμε σε ένα σύστημα υποβοήθησης *(recommendation system)* που θα προτείνει κατάλληλες στρατηγικές για την βελτίωση της απόδοσης των παικτών. Για την υλοποίηση του συστήματος αυτού μπορούμε να χρησιμοποιήσουμε μεθόδους Μηχανιστικής Μάθησης (*Machine Learning* - ML) [Beck 1997].

Οι βασικές αρχές ενός συστήματος υποβοήθησης στηρίζονται σε πρώτη φάση στην συλλογή πληροφορίας για τη συμπεριφορά και την ιδιοσυγκρασία του παίκτη. Ο τρόπος με τον οποίο συλλέγεται αυτή η πληροφορία καθώς και η ποιότητα της είναι σημαντικός. Θέλουμε η συλλογή της πληροφορίας να είναι διάφανη για τον τελικό χρήστη (προκειμένου να μην τον επιβαρύνει με την διαδικασία να δώσει ο ίδιος τις πληροφορίες). Από την άλλη, πληροφορίες που συλλέγονται αυτόματα διέπονται από τον κίνδυνο να μην είναι απόλυτα ακριβείς.

Στη συνέχεια καθοριστικό ρόλο παίζει η οργάνωση της πληροφορίας προκειμένου να παραχθεί γνώση. Αλγόριθμοι μηχανιστικής μάθησης μπορούν να χρησιμοποιηθούν σε αυτήν την φάση για να παράγουν χρήσιμα αποτελέσματα από πληροφορίες του πραγματικού κόσμου (μεγάλη ποσότητα πληροφορίας, θόρυβος, άγνωστη πληροφορία).

Τα αποτελέσματα μπορούν να χρησιμοποιηθούν είτε από ένα κεντρικό πρόσωπο το οποίο θα οργανώσει τη συνέχεια της εκπαιδευτικής διαδικασίας είτε από έναν ηλεκτρονικό πράκτορα *(agent)*.

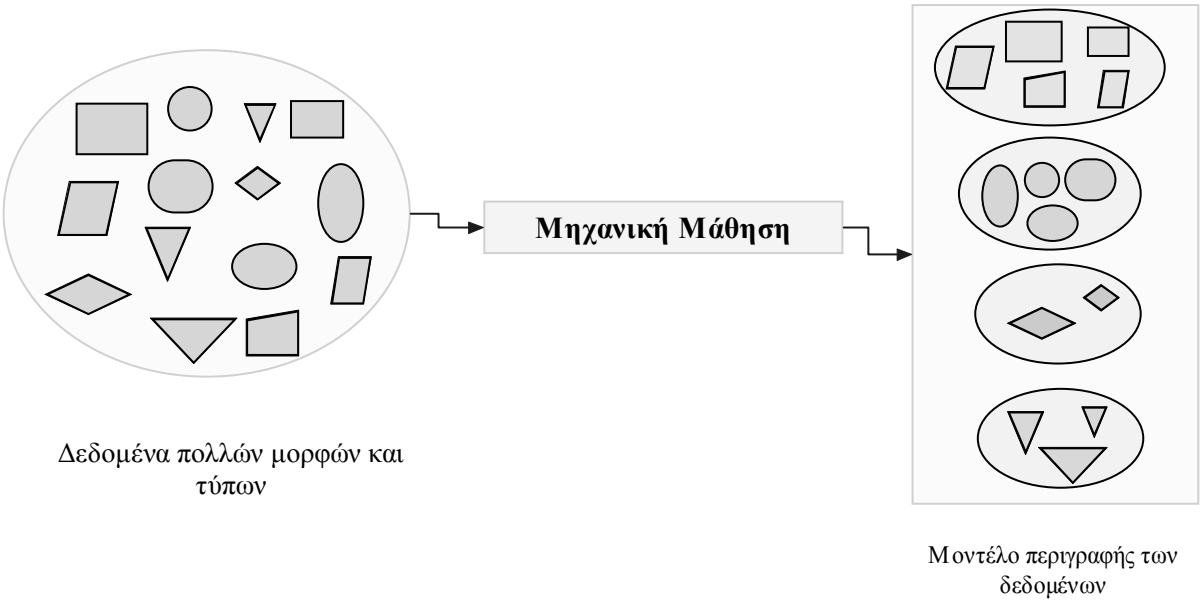
Στο σχήμα 6.1 παρουσιάζεται μια πρώτη ιδέα για τα χαρακτηριστικά του προτεινόμενου συστήματος.



Σχήμα 6.1 Η βασική δομή ενός συστήματος υποβοήθησης της εκπαιδευτικής διαδικασίας

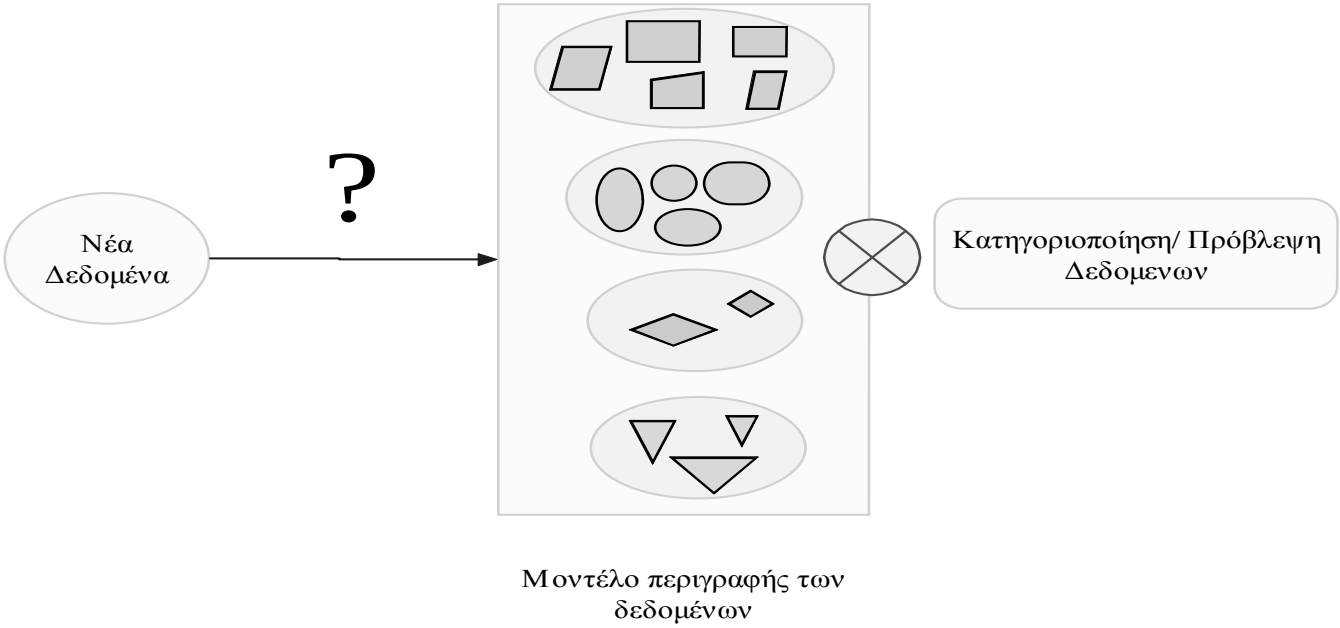
6.2. Ο ρόλος της Μηχανικής Μάθησης

Η Μηχανική Μάθηση *(Machine Learning)* χρησιμοποιεί ακατέργαστα δεδομένα (δηλαδή μεγάλες ποσότητες δεδομένων, με θόρυβο και άγνωστα στοιχεία) για να φτιάξει μοντέλα περιγραφής και ταξινόμησης τους (Σχήμα 6.2).



Σχήμα 6.2 Ο ρόλος της Μηχανικής Μάθησης

Στην περίπτωσή μας θα μπορούσαμε να χρησιμοποιήσουμε κάποια τέτοια τεχνική για να μοντελοποιήσουμε τους παίκτες. Εν συνεχεία, χρησιμοποιώντας αυτά τα μοντέλα μπορούμε να κατηγοριοποιήσουμε τους παίκτες και να προβλέψουμε τη συμπεριφορά τους. (Σχήμα 6.3)



Σχήμα 6.3 Κατηγοριοποίηση και πρόβλεψη δεδομένων μέσω της Μηχανικής Μάθησης

6.3. Παραδείγματα συστημάτων υποβοήθησης (*recommendation system)*

Υπάρχουν αρκετά συστήματα υποβοήθησης (recommendation systems) που αντανακλούν κάποιες από τις ιδιότητες που θέλουμε να υπάρχουν στο σύστημά μας. Για παράδειγμα, υπάρχουν στο διαδίκτυο εφαρμογές που προτείνουν στους χρήστες νέες δικτυακές σελίδες ή υπηρεσίες που πιθανόν να τους αρέσουν ή να τους φανούν χρήσιμες. Οι εφαρμογές αυτές ανακαλύπτουν τις προτιμήσεις του κάθε χρήστη από προηγούμενες επιλογές του ή συγκρίνουν τους χρήστες μεταξύ τους και όταν αυτοί μοιάζουν να έχουν κοινά στοιχεία, τότε στέλνουν τις προτιμήσεις του ενός και στον άλλο.

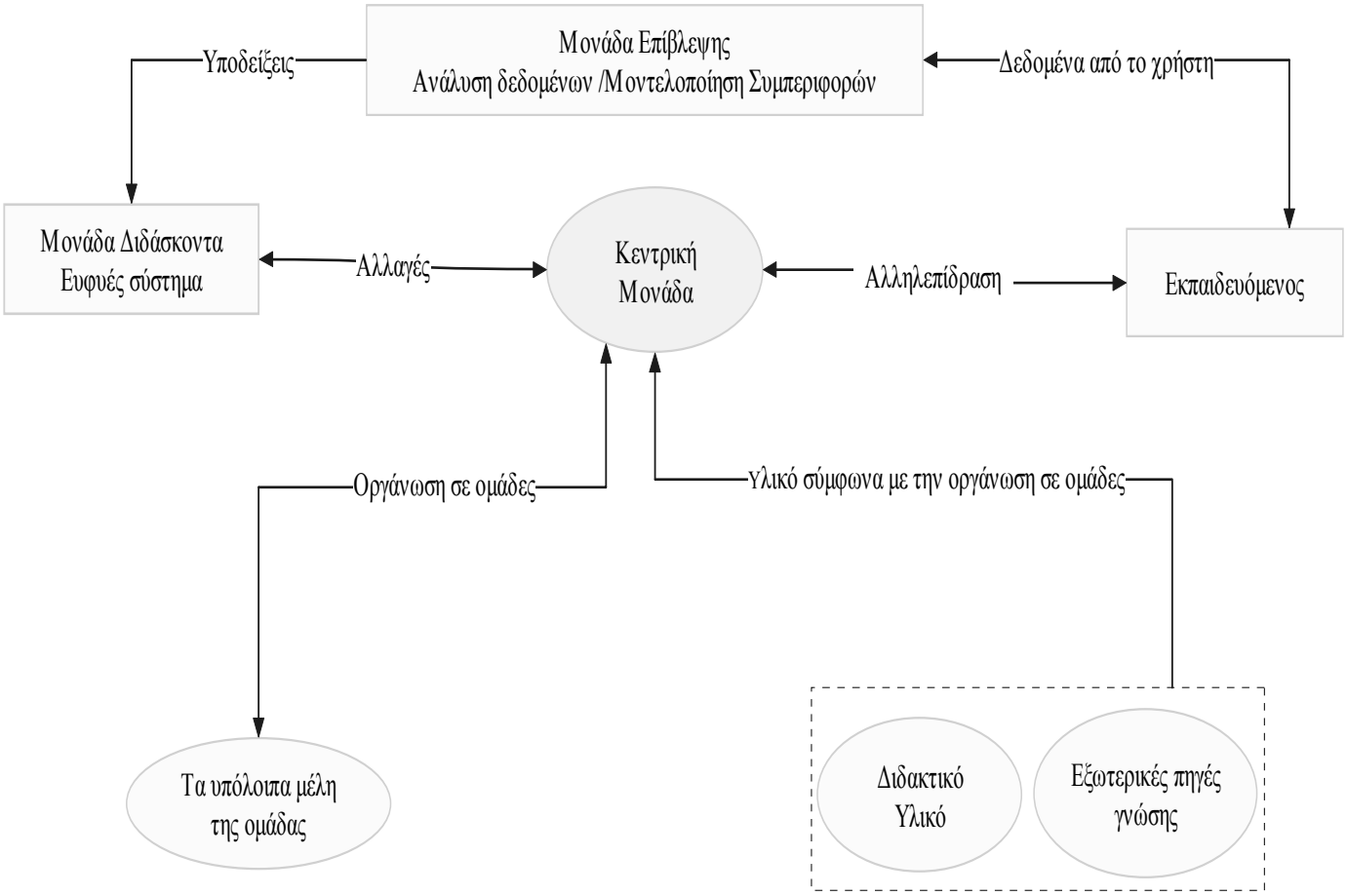
Στη συνέχεια περιγράφουμε κάποια από τα υπάρχοντα συστήματα υποβοήθησης προκειμένου να κατανοήσουμε τις βασικές ιδέες που τα διέπουν.

Ο InfoFinder [Krulwich 1997] δημιουργεί τα προφίλ των χρηστών *(user profiles)* αναλύοντας τα κείμενα που οι χρήστες βρήκαν ενδιαφέροντα. Για κάθε χρήστη δημιουργεί δέντρα απόφασης για τις κατηγορίες που αυτός βρήκε ενδιαφέρουσες και κατασκευάζει εξειδικευμένα queries για θέματα που πιθανόν να ενδιαφέρουν το χρήστη τα οποία και χρησιμοποιεί σαν είσοδο για τις υπάρχουσες search engines.

Παρόμοια είναι και η προσέγγιση που χρησιμοποιείται από το σύστημα PHOAKS [Terveen 1997], το οποίο όμως προχωράει ένα βήμα παραπέρα και κατασκευάζει κοινότητες χρηστών βάσει των ενδιαφερόντων τους. Όταν κάποιος χρήστης βρίσκει μια σελίδα ενδιαφέρουσα αυτή του η επιλογή μεταδίδεται και στους υπόλοιπους χρήστες της κοινότητας.

Το σύστημα FAB [Balavanovic 1997] δημιουργεί κοινότητες χρηστών στις οποίες εγκαθιστά ηλεκτρονικούς πράκτορες *(agents)* που μπορούν να προσαρμοσθούν στις απαιτήσεις των χρηστών. Οι πράκτορες αυτοί αλληλεπιδρούν μεταξύ τους αλλά και με το διαδίκτυο προκειμένου να παράγουν καλύτερα αποτελέσματα. Το σύστημα χρησιμοποιεί πολλές τεχνικές Τεχνητής Νοημοσύνης και ένα client-server μοντέλο για την εξόρυξη της γνώσης.

Η αρχιτεκτονική ενός συστήματος υποβοήθησης που χρησιμοποιεί Μηχανική Μάθηση θα μπορούσε να μοιάζει με αυτή του σχήματος 6.4.



Σχήμα 6.4 Η αρχιτεκτονική ενός συστήματος υποβοήθησης

6.4. Πλεονεκτήματα της μοντελοποίησης

Η μοντελοποίηση των παικτών συντελεί στη μακροπρόθεσμη βελτίωση τους όσον αφορά στον τρόπο με τον οποίο παίζουν. Πιο συγκεκριμένα μέσω της μοντελοποίησης:

- Εξάγουμε συμπεράσματα για το γνωστικό επίπεδο, τις ικανότητες και τα αδύνατα σημεία των παικτών και προσαρμόζουμε ανάλογα τις συμβουλές που τους δίνουμε. Γνωστοποιώντας τα συμπεράσματα στους ίδιους τους παίκτες τους βοηθάμε να βελτιώσουν τη στρατηγική τους και να αντιμετωπίσουν τα αδύνατα σημεία τους. Το μοντέλο δηλαδή χρησιμοποιείται σαν ένας ειδικός, ικανός να βοηθήσει τον παίκτη να αποκτήσει μια πλήρη εικόνα του επιπέδου του.
- Ο κάθε παίκτης μπορεί να συγκρίνει την απόδοσή του σε σχέση με το νευρωνικό (το νευρωνικό παίζει το ρόλο του δασκάλου - κάνουμε την παραδοχή ότι το νευρωνικό έχει εκπαιδευτεί καλά και είναι ικανός δάσκαλος) προκειμένου να γνωρίζει ανά πάσα στιγμή τη συγκριτική τους θέση.
- Το μοντέλο καταγράφει τις προτιμήσεις του παίκτη ως προς τη στρατηγική που ακολουθεί, τη συμπεριφορά του στο παιχνίδι και άλλες παραμέτρους που επηρεάζουν τη μάθηση. Αντιμετωπίζει κάθε παίκτη ως μοναδικό και οι συμβουλές του είναι εξατομικευμένες και προσαρμοσμένες στον εκάστοτε παίκτη. Το γεγονός αυτό πέραν του ότι ενισχύει την κριτική ικανότητα, την ανεξαρτησία και την αυτονομία του παίκτη αυξάνει επίσης και το ενδιαφέρον του για το παιχνίδι.

- Μπορούμε να δημιουργήσουμε κοινότητες παικτών (που θα αποτελούνται από παίκτες με παρόμοια χαρακτηριστικά και ενδιαφέροντα) με σκοπό τη συνεργασία, την αλληλοβοήθεια και την ανταλλαγή απόψεων σχετικά με το παιχνίδι. Έτσι το σύστημα μπορεί να παρέχει σε παίκτες με παρόμοια μοντέλα παραπλήσιες συμβουλές και προτάσεις.

Μειώνεται ο χρόνος της εκπαίδευσης του κάθε παίκτη, καθώς μέσω του μοντέλου έχουμε σαφή εικόνα της κατάστασης του και γνωρίζουμε σε ποια ακριβώς σημεία χρειάζεται περαιτέρω βελτίωση. Στην περίπτωση μάλιστα των κοινοτήτων παικτών ο χρόνος μειώνεται σε σημαντικό βαθμό καθώς κάποιοι παίκτες μπορεί να έχουν παρόμοια μοντέλα.

6.5. Τεχνικές μοντελοποίησης

Υπάρχουν αρκετές τεχνικές για την μοντελοποίηση των στοιχείων που αφορούν τον παίκτη και την μάθησή του κατά τη διάρκεια του παιχνιδιού [Baffes 1994], [Baffes 1996], [Baffes, Mooney 1996]. Ο παίκτης αντιμετωπίζεται σαν ένας μαθητής *(learner)* και στόχος της εκπαιδευτικής διαδικασίας είναι η βελτίωση της στρατηγικής του σε μακροπρόθεσμο επίπεδο. Στη συνέχεια αναπτύσσουμε πέντε τεχνικές μοντελοποίησης και αναφέρουμε τα πλεονεκτήματα και τα μειονεκτήματά τους:

1. Μοντελοποίηση επίστρωσης *(Overlay modeling)*

Στο μοντέλο επίστρωσης *(overlay model)* η γνώση του παίκτη αποτελεί υποσύνολο της πραγματικής γνώσης του πεδίου. Καθώς ο παίκτης εκτελεί ενέργειες που υποδηλώνουν ότι κατανοεί συγκεκριμένα γνωστικά αντικείμενα του πεδίου, αυτά μαρκάρονται στο μοντέλο επίστρωσης. Πιο εξελιγμένα μοντέλα της κατηγορίας αυτής, εκφράζουν και το ποσοστό στο οποίο ο παίκτης κατέχει ένα τέτοιο γνωστικό αντικείμενο. Τυπικά, αυτά που δεν έχουν μαρκαριστεί αντιπροσωπεύουν άγνωστα προς τον παίκτη αντικείμενα μάθησης ή ανεπαρκώς κατανοημένα και χρησιμεύουν ως «οδηγοί» στην πορεία της εκπαιδευτικής διαδικασίας. Αυτό που θέλουμε είναι η γνώση του παίκτη να πλησιάσει τη γνώση του πεδίου.

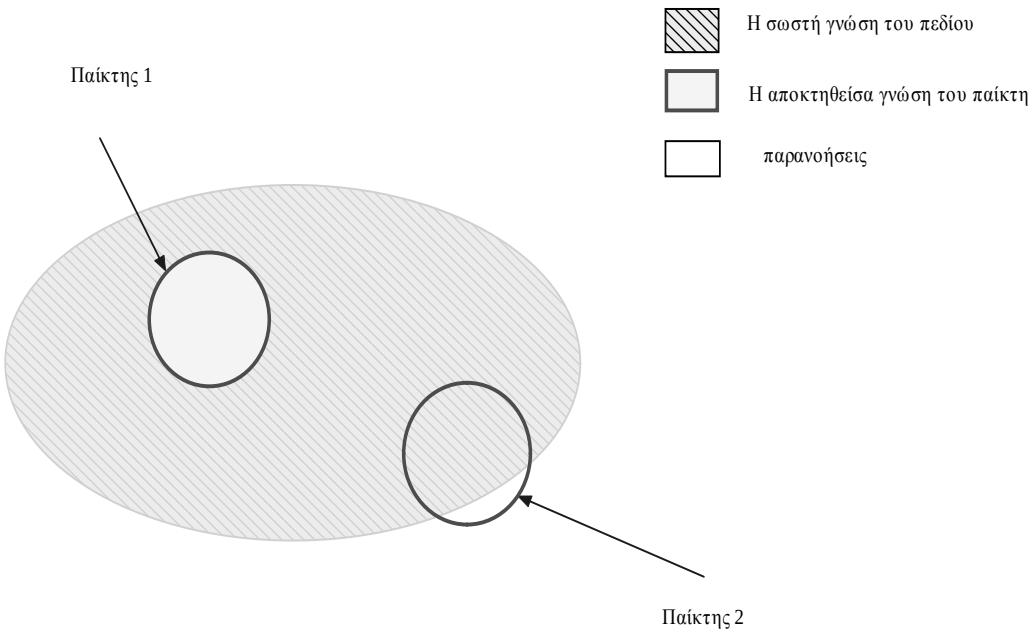
Πλεονεκτήματα:

- Εύκολη υλοποίηση

Μειονεκτήματα:

- Το μοντέλο είναι στατικό και δεν μοντελοποιεί τυχόν μη αναμενόμενες ενέργειες του παίκτη.
- Δεν λαμβάνει υπόψη το γεγονός ότι ο παίκτης μπορεί να έχει παρανοήσει κάποιο στοιχείο του παιχνιδιού (λανθασμένη γνώση - παρανόηση).

Στην περίπτωση του παιχνιδιού μας ως σωστή γνώση του πεδίου θα μπορούσε να θεωρηθεί η γνώση του νευρωνικού συστήματος και ως αποκτηθείσα γνώση του παίκτη θα μπορούσαν να θεωρηθούν π.χ. κινήσεις που προσεγγίζουν τη βάση του αντιπάλου



Σχήμα 6.4 Παράδειγμα ενός μοντέλου επίστρωσης (overlay model)

2. Μοντελοποίηση των λαθών *(Bug library)*

Στη μοντελοποίηση των λαθών το μοντέλο του παίκτη δημιουργείται συγκρίνοντας τις ενέργειες του παίκτη με έναν κατάλογο αναμενόμενων λαθών *(bug library)*. Ο κατάλογος αυτός κατασκευάζεται εκ των προτέρων με το χέρι μέσω της ανάλυσης των λαθών των παικτών. Το μοντέλο αυτό επεκτείνει το χώρο των λαθών που ενδέχεται να κάνει ο παίκτης.

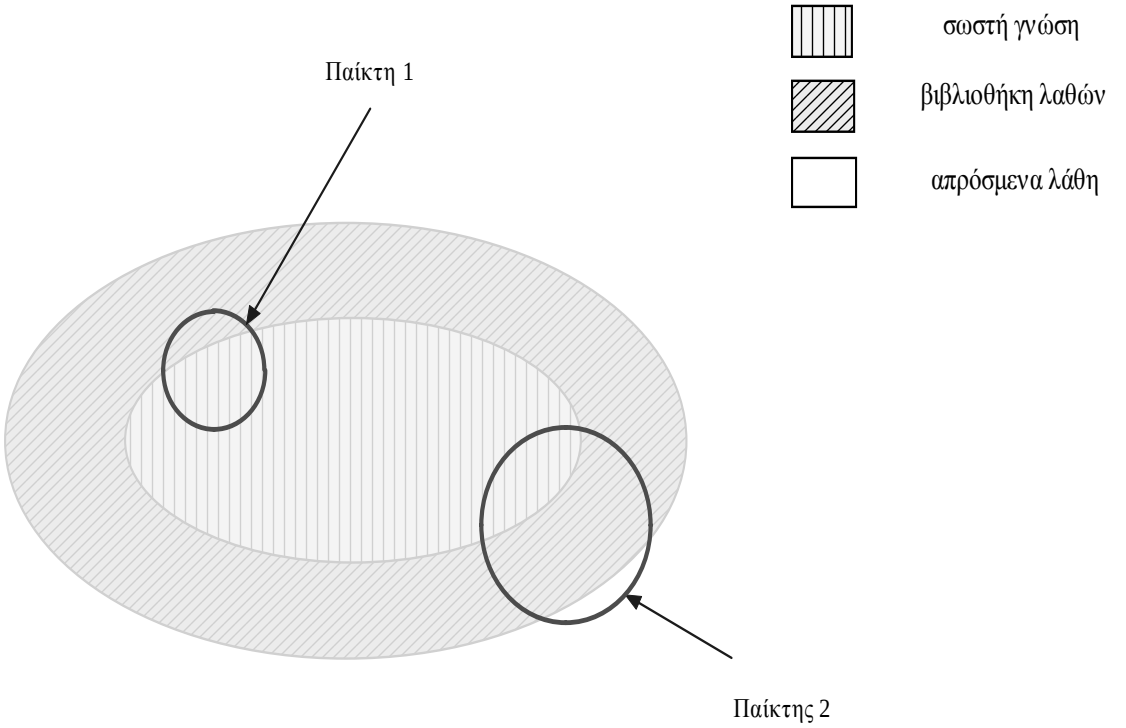
Πλεονεκτήματα:

- Το μοντέλο λαμβάνει υπόψη το γεγονός ότι ο παίκτης μπορεί να έχει μάθει κάτι λάθος.

Μειονεκτήματα:

- Η δημιουργία και η συντήρηση του καταλόγου των λαθών είναι δύσκολη και χρονοβόρα.
- Το μοντέλο είναι στατικό, συνεπώς δε μπορεί να “συλλάβει” το σύνολο της συμπεριφοράς του παίκτη που μεταβάλλεται διαρκώς με το χρόνο.

Στην περίπτωση του παιχνιδιού μας ο κατάλογος των αναμενόμενων λαθών θα μπορούσε να περιλαμβάνει κινήσεις μαζικής εξόδου από τη βάση (η κίνηση αυτή είναι λανθασμένη γιατί οδηγεί στην απώλεια όλων των υπολοίπων πιονιών του παίκτη), κινήσεις προσέγγισης των πιονιών του αντιπάλου (η κίνηση αυτή δεν είναι καλή διότι εμπεριέχει τον κίνδυνο της απώλεια πιονιών του παίκτη) και άλλες περιπτώσεις κινήσεων.



Σχήμα 6.5 Παράδειγμα μιας βιβλιοθήκης λαθών (bug library)

3. Δυναμική μοντελοποίηση των λαθών *(Dynamic modeling of bugs)*

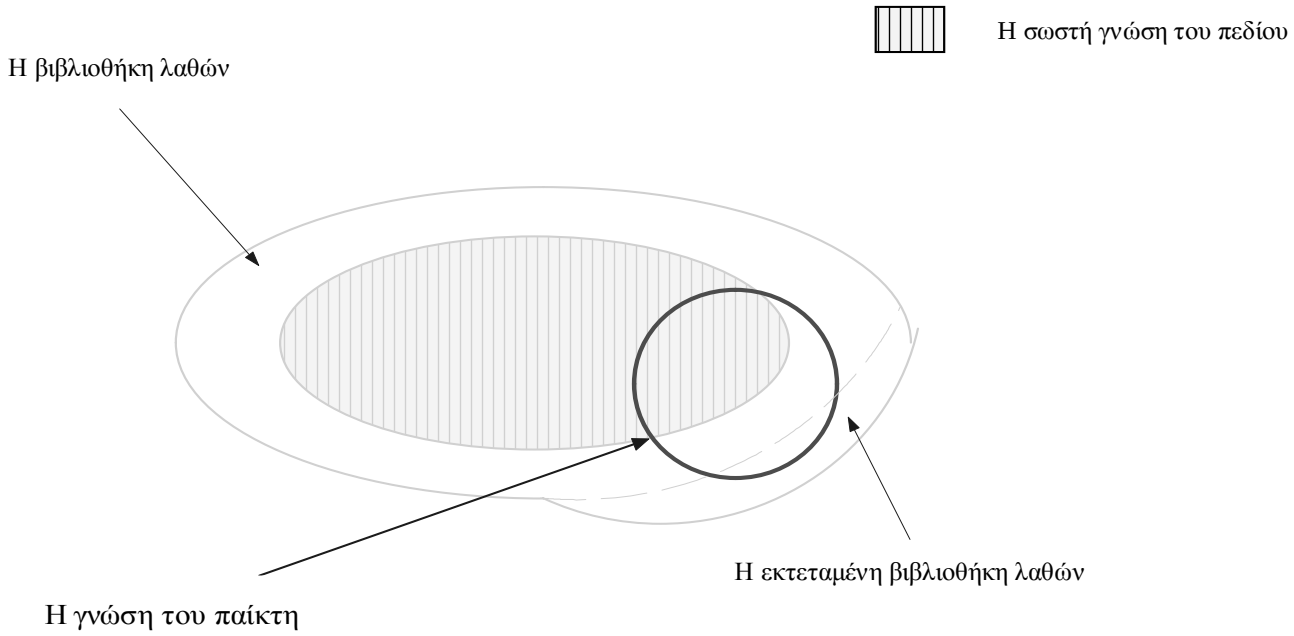
Στη δυναμική μοντελοποίηση των λαθών *(dynamic modeling of bugs)* η βιβλιοθήκη των λαθών *(bug library)* επεκτείνεται και νέες πληροφορίες για τα λάθη που ενδέχεται να κάνει ο παίκτης κατασκευάζονται δυναμικά. Αυτό επιτυγχάνεται μέσω μιας αναλυτικής προσέγγισης που στηρίζεται σε ένα μηχανισμό που εξετάζει πως οι μέχρι στιγμής γνωστές παρανοήσεις του παίκτη μπορούν να γενικευτούν ώστε να είναι εφικτή η επιτυχής πρόβλεψη τυχόν μελλοντικών του λαθών. Αυτά τα αναμενόμενα λάθη στη συνέχεια παρουσιάζονται στον «εκπαιδευτή», ο οποίος και αποφασίζει αν θα τα εντάξει τελικά στην εκπαιδευτική διαδικασία.

Πλεονεκτήματα:

- Το μοντέλο είναι δυναμικό με αποτέλεσμα να “συλλαμβάνει” οποιαδήποτε αλλαγή στον τρόπο με τον οποίο παίζει ο παίκτης.

Μειονεκτήματα:

- Το μοντέλο δεν είναι αυτόνομο, ο «εκπαιδευτής» είναι αυτός που τελικά αποφασίζει αν θα εντάξει τα λάθη που προέβλεψε το σύστημα στη διαδικασία εκπαίδευσης του παίκτη.



Σχήμα 6.6 Παράδειγμα επέκτασης μιας βιβλιοθήκης λαθών (dynamic modeling of bugs)

4. Μοντελοποίηση των παρανοήσεων από το μηδέν *(Modeling misconception from scratch)*

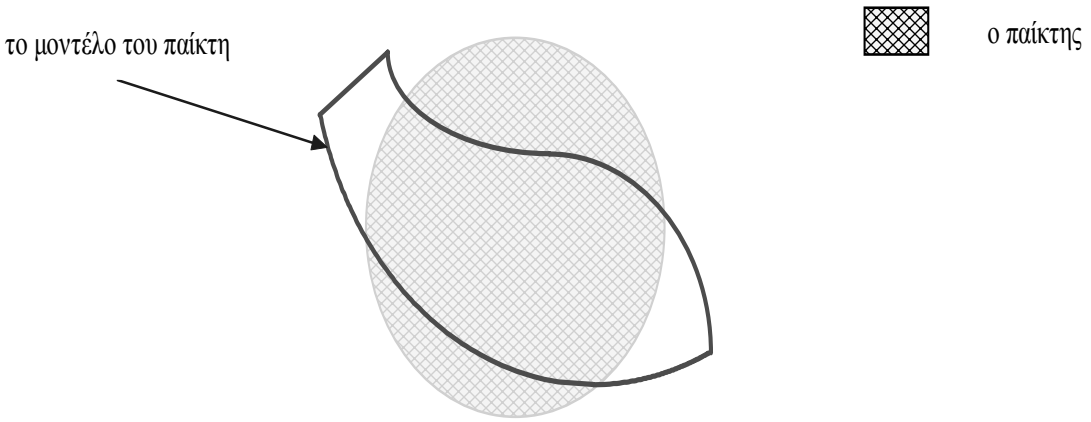
Στο μοντέλο των παρανοήσεων από το μηδέν *(modeling misconceptions from scratch)*, η μοντελοποίηση του παίκτη ξεκινά από το μηδέν και εν συνεχεία κατασκευάζονται δυναμικά νέες πληροφορίες για τα πιθανά μελλοντικά του λάθη. Προκειμένου να εξαχθούν τα πιο αντιπροσωπευτικά στοιχεία του μοντέλου του παίκτη από τα διάφορα στιγμιότυπα της συμπεριφοράς του, χρησιμοποιείται η τεχνική της αναγωγής. Στην αρχή το σύστημα δεν γνωρίζει τίποτα για τον παίκτη, αλλά στην πορεία παρατηρώντας τον τρόπο με τον οποίο παίζει κατασκευάζει ένα πλήρες μοντέλο του. Στο μοντέλο αυτό παράλληλα με τα γνωστά στον παίκτη στοιχεία του παιχνιδιού (σωστή γνώση) ενσωματώνονται και στοιχεία που ο παίκτης έχει παρανοήσει (λανθασμένη γνώση).

Πλεονεκτήματα:

- Το μοντέλο είναι δυναμικό.
- Το μοντέλο είναι αυτόνομο και δεν απαιτείται για την αναγωγή η καθοδήγηση από κάποιον «εκπαιδευτή».

Μειονεκτήματα:

- Απαιτείται ένα μεγάλο πλήθος παραδειγμάτων εκπαίδευσης για να μπορούν να γίνουν ακριβείς προβλέψεις για τη συμπεριφορά του παίκτη.



Σχήμα 6.7 Παράδειγμα μοντελοποίησης των λαθών από το μηδέν *(Modeling student misconception from scratch)*

5. Βελτίωση της θεωρίας *(Theory refinement)*

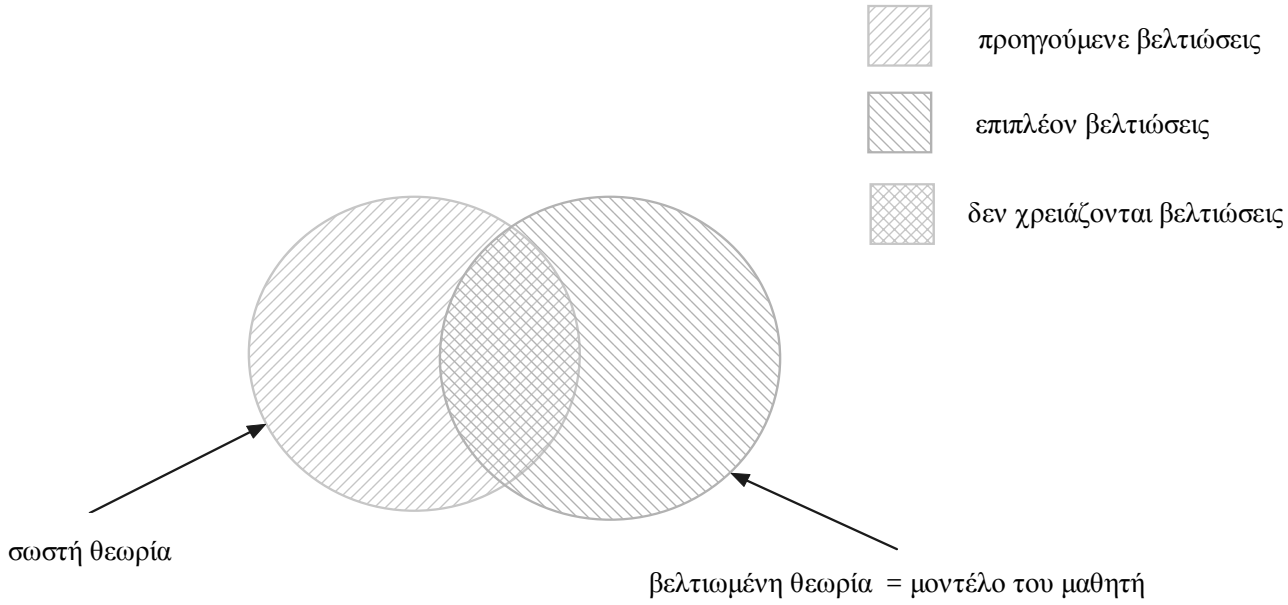
Στο μοντέλο της βελτίωσης της θεωρίας *(theory refinement)* το επίπεδο των γνώσεων του παίκτη ανανεώνεται δυναμικά ώστε να είναι συμβατό με τα παραδείγματα μάθησης που του παρουσιάζει το σύστημα. Στην πράξη, η γνώση του παίκτη μπορεί να είναι μικρή ή ανεπαρκής. Τα παραδείγματα μάθησης αντιπροσωπεύουν τη συμπεριφορά που θα πρέπει να επιδείξει το μοντέλο του παίκτη στην αντιμετώπιση διαφόρων καταστάσεων. Σκοπός της τεχνικής αυτής είναι να βρει ένα μοντέλο το οποίο θα αναπαράγει ακριβώς τη συμπεριφορά που περιγράφεται μέσω των παραδειγμάτων. Η ιδέα στηρίζεται στην σταδιακή τροποποίηση, δηλαδή αρχικά ο παίκτης γνωρίζει κάποια πράγματα για το παιχνίδι που ενδέχεται όμως να μην είναι απόλυτα σωστά. Στη συνέχεια το σύστημα τροποποιεί την ήδη υπάρχουσα γνώση του παίκτη ώστε να συμβαδίζει με τα παραδείγματα μάθησης. Αν τα παραδείγματα μάθησης αντιπροσωπεύουν τη συμπεριφορά που το σύστημα θέλει να διδάξει στον παίκτη, τότε όλες οι τροποποιήσεις που γίνονται στο μοντέλο του παίκτη έχουν ως σκοπό να επισημάνουν τις τυχόν παρανοήσεις του και να τις εξουδετερώσουν.

Πλεονεκτήματα:

- Το μοντέλο αντιμετωπίζει τα παραδείγματα μάθησης με έναν ενιαίο τρόπο με αποτέλεσμα η διαδικασία μάθησης να είναι ανεξάρτητη του γνωστικό αντικειμένου.
- Το μοντέλο αξιοποιεί την τυχόν ήδη υπάρχουσα γνώση του παίκτη με αποτέλεσμα η αναγωγή της συμπεριφοράς του παίκτη από τα παραδείγματα μάθησης να λαμβάνει λιγότερο χρόνο.
- Το μοντέλο έχει τη δυνατότητα να εντοπίζει τα σημεία που ο παίκτης έχει αντιληφθεί λάθος (παρανοήσεις).
- Το μοντέλο χρησιμοποιεί για την εκπαίδευση του παίκτη παραδείγματα μάθησης που θεωρεί ότι είναι κατάλληλα για να μάθει ο παίκτης συγκεκριμένα στοιχεία του παιχνιδιού.
- Η εκπαιδευτική διαδικασία προχωρά ακόμη και αν ο παίκτη δε γνωρίζει τίποτα για το παιχνίδι (βέβαια η μάθηση συντελείται με πιο αργούς ρυθμούς).

Μειονεκτήματα:

- Το σύστημα δεν μπορεί να ελέγξει αν τα παραδείγματα μάθησης μέσω των οποίων εκπαιδεύεται ο παίκτης είναι σωστά.



Σχήμα 6.8 Παράδειγμα μοντελοποίησης μέσω βελτίωσης της θεωρίας *(theory refinement)*

6.6. Τι δεδομένα χρειαζόμαστε για τη μοντελοποίηση των παικτών

Είναι δύσκολο να αποκτήσουμε ακριβή εικόνα της συμπεριφοράς ενός παίκτη κατά τη διάρκεια του παιχνιδιού. Ωστόσο υπάρχουν στοιχεία της συμπεριφοράς αυτής που θα μπορούσαμε να τα «μετρήσουμε» και βάσει αυτών να βγάλουμε συμπεράσματα που θα τα χρησιμοποιήσουμε για τη βελτίωση του. Τα στοιχεία αυτά μπορεί να είναι είτε απλά στατιστικά στοιχεία είτε πιο σύνθετα συμπεράσματα, κανόνες και συμβουλές σχετικά με τα μέτρα που θα πρέπει να ληφθούν προκειμένου ο παίκτης να βελτιωθεί. Στην περίπτωση του παιχνιδιού μας τέτοια στοιχεία θα μπορούσαν να αφορούν:

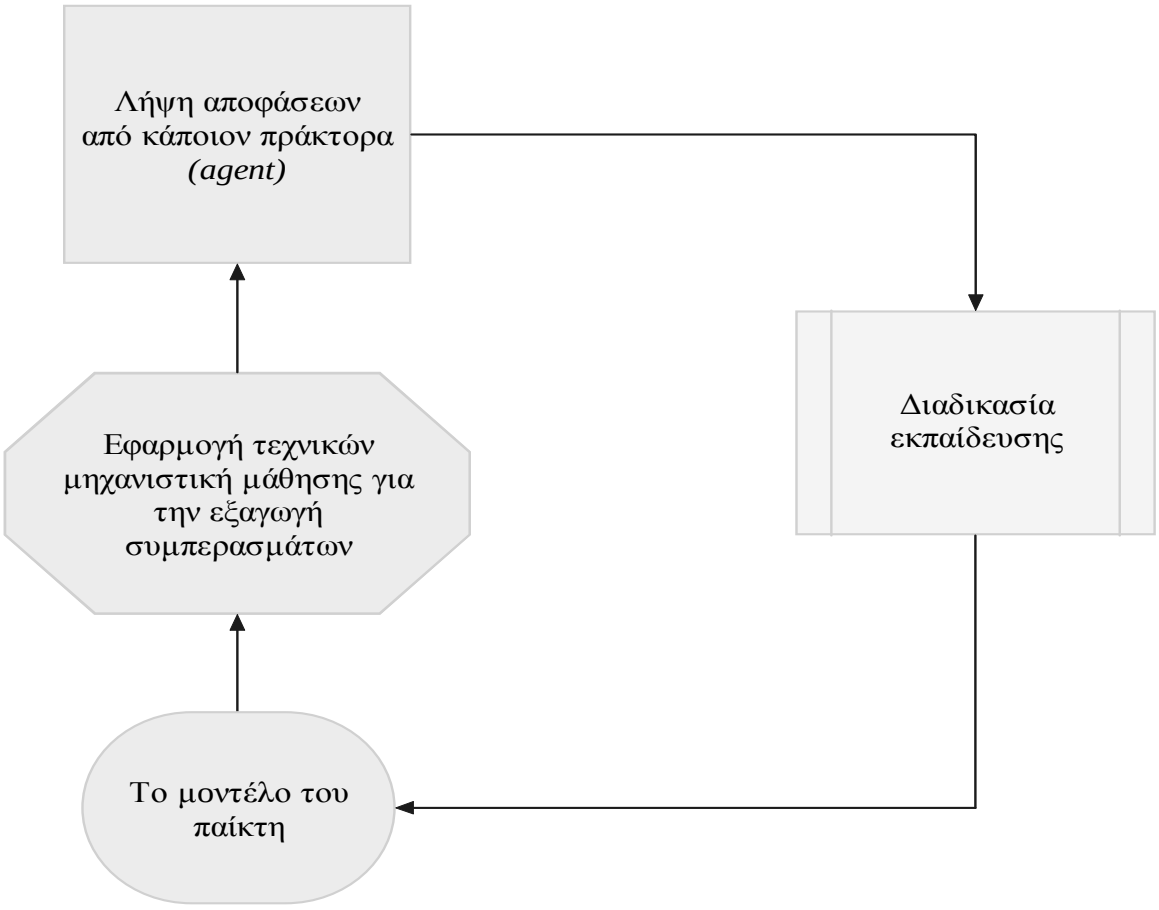
- την απόδοση του παίκτη σε κάθε παιχνίδι (κέρδισε ή έχασε).
- την αξία των κινήσεων του παίκτη (η αξιολόγηση κάθε κίνησης μπορεί να γίνει με τεχνικές Ενισχυμένης Μάθησης - RL).
- το πλήθος των ενεργών πιονιών του παίκτη κατά τη διάρκεια του παιχνιδιού
- το πλήθος των ενεργών πιονιών του παίκτη που υπάρχουν στη βάση του κατά τη διάρκεια του παιχνιδιού.
- το πλήθος των κινήσεων ανά παιχνίδι.
- ο χρόνος μεταξύ διαδοχικών κινήσεών του.
- τη συμπεριφορά του παίκτη σε σχέση με τις κινήσεις που του προτείνει το νευρωνικό σύστημα (κάνουμε την παραδοχή ότι η γνώση του νευρωνικού είναι σωστή).

Επισημαίνουμε και πάλι πως το να αποκτήσουμε ακριβή εικόνα του παίκτη είναι αδύνατο. Για το λόγο αυτό εξάλλου, τα περισσότερα μοντέλα είναι προσεγγιστικά, κάτι που οπωσδήποτε θα πρέπει να το

λάβουμε σοβαρά υπόψη μας στην παραπέρα δρομολόγηση της διαδικασίας εκπαίδευσης του παίκτη. Ένας άλλος περιορισμός έγκειται και στην προσεγγιστική φύση του νευρωνικού, δεν μπορούμε να είμαστε σίγουροι ότι η γνώση που απέκτησε το νευρωνικό κατά την εκπαίδευσή του είναι πάντα σωστή.

6.7. Πως λαμβάνονται οι αποφάσεις για την πορεία της εκπαιδευτικής διαδικασίας

Σε ένα κλασσικό σύστημα εκπαίδευσης παικτών ο «εκπαιδευτικός» είναι αυτός που αποφασίζει για θέματα που άπτονται της διαδικασίας εκπαίδευσης συμβουλευόμενος το μοντέλο του παίκτη. Πέραν όμως του εκπαιδευτικού, συμπεράσματα τέτοιου είδους είναι εύκολο να εξαχθούν με χρήση διαφόρων τεχνικών μηχανιστικής μάθησης (machine learning) όπως η ενισχυμένη μάθηση, τα δέντρα απόφασης, οι γενετικοί αλγόριθμοι, τα νευρωνικά δίκτυα κ.α. Προς αυτή την κατεύθυνση μια ενδιαφέρουσα προοπτική θα ήταν το ίδιο το σύστημα να αποφασίζει για την διαδικασία εκπαίδευσης. Δηλαδή δεδομένου του μοντέλου του παίκτη να υπάρχει ένας μεσάζοντας *(agent)*, ο οποίος και θα λαμβάνει αποφάσεις που θα συντελούν στη βελτίωση του παίκτη. Σχηματικά η λήψη των αποφάσεων θα μπορούσε να απεικονιστεί ως εξής:



Σχήμα 6.9 Λήψη αποφάσεων για τη βελτίωση του παίκτη

6.8. Μοντελοποίηση και Ενισχυτική Μάθησης (RL)

Η μηχανιστική μάθηση «κρίνει» ποια στοιχεία του παιχνιδιού είναι γνωστά στον παίκτη και ποια όχι και εξετάζει τους λόγους για τους οποίους ο παίκτης κάνει σφάλματα. Το ζητούμενο είναι το σύστημα να συσχετίζει τις ενέργειες του «εκπαιδευτικού» με το γνωστικό επίπεδο των παικτών.

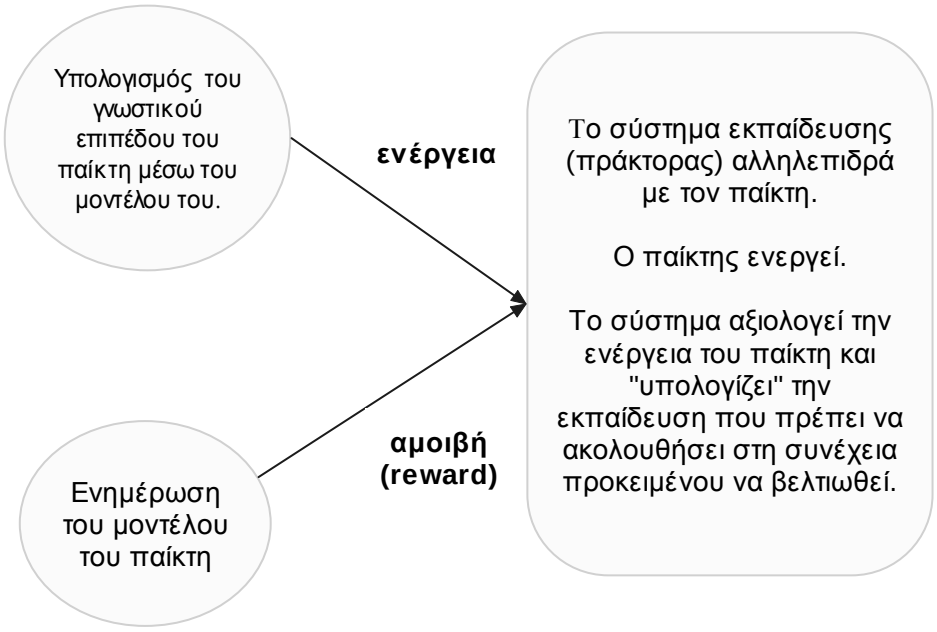
Στην κορυφή του μοντέλου του παίκτη υπάρχει ένας πράκτορας *(learning agent)*, ο οποίος μαθαίνει κατά πόσο οι αλληλεπιδράσεις του παίκτη με το σύστημα βελτιώνουν τις γνωστικές του ικανότητες. Οι πληροφορίες που συλλέγονται με τον τρόπο αυτό είναι πολλές, καθώς το σύστημα προσπαθεί να αποκτήσει μία γενικότερη εικόνα της γνώσης του παίκτη και δεν αναλώνεται σε ένα μόνο συγκεκριμένο πράγμα. Χρειάζεται βέβαια μεγάλη προσοχή στην αναπαράσταση της γνώσης καθώς οι είσοδοι που θα δοθούν στον πράκτορα θα πρέπει να είναι της μορφής: “η γνώση του παίκτη για το θέμα”, “η απόδοση του παίκτη όσον αφορά στο συγκεκριμένο θέμα” κ. ο. κ.

Η ενισχυτική μάθηση είναι κυρίως μία μέθοδος αξιολόγησης του επιπέδου του παίκτη όσον αφορά τις ικανότητές του στο παιχνίδι και ανάθεσης σ’ αυτόν επαίνου ή μομφής, ανάλογα με το πόρισμα της αξιολόγησης. Στόχος του παίκτη είναι η όσο το δυνατόν μεγαλύτερη συνολική επιβράβευση.

Δύο είναι τα βασικά συστατικά ενός συστήματος που χρησιμοποιεί ενισχυτική μάθηση:

- Το περιβάλλον, δηλαδή οτιδήποτε υπάρχει πέρα από τον άμεσο έλεγχο του παίκτη.
- Οι ενέργειες, δηλαδή όλες οι πληροφορίες που συλλέγει ο πράκτορας.

Όπως φαίνεται και στο ακόλουθο σχήμα ο μεσάζων αφού εξετάσει τη γνώση του παίκτη (μέσω του μοντέλου του) επιλέγει ποια ενέργεια θα εκτελέσει. Στο σημείο αυτό το περιβάλλον αναλαμβάνει τον έλεγχο και οι επιδράσεις της παραπάνω ενέργειας γίνονται εμφανείς. Βάσει της νέας κατάστασης του παίκτη δίνεται μια αμοιβή στον πράκτορα. Στόχος του παίκτη είναι να μεγιστοποιήσει αυτές τις αμοιβές. Από τη στιγμή που η αμοιβή δόθηκε η προηγούμενη κατάσταση του παίκτη θα πρέπει να αντικατασταθεί από τη νέα κατάσταση.



Σχήμα 6.10 Αλληλεπίδραση agent - παίκτη

6.9. Αξιολόγηση της χρήσης RL για τη μοντελοποίηση των παικτών

Στη συνέχεια αναφέρουμε επιγραμματικά κάποια από τα πλεονεκτήματα και μειονεκτήματα της μοντελοποίηση των παικτών μέσω τεχνικών Ενισχυτικής Μάθησης (RL).

Πλεονεκτήματα:

- Μπορούμε να λαμβάνουμε υψηλού επιπέδου στρατηγικές αποφάσεις, ακόμη και αν οι πληροφορίες που παρέχονται από το μοντέλο του παίκτη είναι χαμηλού επιπέδου.
- Μειώνεται το κόστος κατασκευής του συστήματος εκπαίδευσης και βελτιώνεται ο χειρισμός των διαφόρων εκπαιδευτικών στρατηγικών καθώς το σύστημα είναι προσαρμοσμένο στις ανάγκες του κάθε παίκτη.
- Τυχόν δεδομένα εισόδου που εμπεριέχουν θόρυβο δεν δημιουργούν προβλήματα και μπορούμε να χειριζόμαστε με επιτυχία και μη προβλεπόμενες ενέργειες του παίκτη (ας μην ξεχνάμε πως οι παίκτες είναι άνθρωποι και ως εκ τούτου η συμπεριφορά τους μπορεί να αλλάξει ανά πάσα στιγμή).
- Δεν χρειάζεται το σύστημα να γνωρίζει εκ των προτέρων το αντικείμενο μάθησης.
- Είναι πιο εύκολο να δώσουμε στο σύστημα εισόδους που επισπεύδουν την εκπαιδευτική διαδικασία και γενικεύουν πιο εύκολα.
- Μπορούμε να ενημερώνουμε το σύστημα σε πραγματικό χρόνο *(real time update)*

Μειονεκτήματα:

- Απαιτείται μεγάλος αριθμός παραδειγμάτων εκπαίδευσης.
- Η γνώση του «δασκάλου» δεν είναι πάντα σωστή και πλήρης.
- Όλα τα μοντέλα είναι προσεγγιστικά, συνεπώς υπάρχει πάντα ο κίνδυνος το σύστημα να έχει σχηματίσει λάθος εικόνα για το επίπεδο του παίκτη.

6.10. Εργαλεία εξόρυξης δεδομένων (data mining)

Με τον όρο εξόρυξη δεδομένων *(data mining)* αναφερόμαστε στη διαδικασία της ανακάλυψης άγνωστων χρήσιμων προτύπων *(pattern)* σε μεγάλες ποσότητες δεδομένων. Τα πρότυπα αυτά είναι συνήθως οργανωμένα σε κάποιο μοντέλο πρόβλεψης ή κατηγοριοποίησης και η ανακάλυψή τους γίνεται μέσω διαφόρων μεθόδων όπως η μηχανική μάθηση, το fuzzy logic, τα νευρωνικά δίκτυα κι άλλα.

Από τα πιο γνωστά εργαλεία που μπορούν να χρησιμοποιηθούν για την εξόρυξη δεδομένων είναι το εργαλείο Weka, το οποίο θα θέλαμε να χρησιμοποιήσουμε και στην περίπτωσή μας προκειμένου να κατηγοριοποιήσουμε τους παίκτες και να βγάλουμε χρήσιμα συμπεράσματα για τη συμπεριφορά τους.

Το Weka είναι ένα εργαλείο εξόρυξης πληροφορίας που αποτελείται από τους πιο γνωστούς αλγορίθμους Μηχανικής Μάθησης. Είναι υλοποιημένο σε γλώσσα Java και αναπτύχθηκε απ' το Πανεπιστήμιο του Waikato στη Νέα Ζηλανδία. Διατίθεται δωρεάν στη διεύθυνση www.cs.waikato.ac.nz/ml/weka.

Με χρήση του Weka διάφορες μέθοδοι μάθησης μπορούν να εφαρμοστούν σε μία βάση δεδομένων και να εξαχθούν πληροφορίες από τα δεδομένα αυτά. Ή μπορούν να εφαρμοστούν διάφοροι αλγόριθμοι μάθησης, γνωστοί ως ταξινομητές *(classifiers)*, και να συγκριθεί η απόδοσή τους προκειμένου να επιλεχθεί κάποιος απ' αυτούς για την πρόβλεψη άγνωστων δεδομένων εισόδου.

Η είσοδος του Weka είναι συνήθως κάποιο κατάλληλα μορφοποιημένο αρχείο (.arff) που περιλαμβάνει στοιχεία του προβλήματος όπως: το όνομα της βάσης, το πλήθος των στιγμιότυπων *(instances)*, το πλήθος των χαρακτηριστικών *(attributes)*, τις τιμές των χαρακτηριστικών (αριθμητικές, διακριτές), τις πιθανές κλάσεις του προβλήματος, τυχόν άγνωστα δεδομένα *(missing attribute values)*. Στο σχήμα 6.11 φαίνεται η δομή του weather.arff αρχείου που παρέχεται μαζί με το Weka και αφορά καιρικά δεδομένα.

```
@relation weather
@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
```

Πίνακας 6.1 Παράδειγμα αρχείου εισόδου στο σύστημα Weka

Στη συνέχεια επιλέγουμε ποια από τα χαρακτηριστικά του προβλήματος θα χρησιμοποιήσουμε για την κατηγοριοποίηση και ποιον αλγόριθμο ταξινόμησης και ξεκινάμε την κατηγοριοποίηση. Το αποτέλεσμα για το αρχείο weather.arff με χρήση ενός ταξινομητή που χρησιμοποιεί νευρωνικά δίκτυα φαίνεται στο ακόλουθο σχήμα (Πίνακας 6.2).

```
=== Run information ===

Scheme:       weka.classifiers.neural.NeuralNetwork -L 0.3 -M 0.2 -N 500 –V 0 -S 0 -E 20 -H a
Relation:     weather
Instances:    14
Attributes:   5
              Outlook
              Temperature
              Humidity
              Windy
              Play
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

weka.classifiers.neural.NeuralNetwork@4ba9a2

=== Stratified cross-validation ===

=== Summary ===
Correctly Classified Instances          10               71.4286 %
Incorrectly Classified Instances        4               28.5714 %
Kappa statistic                          0.3778
Mean absolute error                      0.3235
Root mean squared error                  0.5081
Relative absolute error                 69.6872 %
Root relative squared error            105.9669 %
Total Number of Instances               14

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   Class
                 0.778    0.4        0.778     0.778     0.778     yes
                 0.6      0.222      0.6       0.6       0.6       no

=== Confusion Matrix ===

 a b   <-- classified as
 7 2 | a = yes
 2 3 | b = no
```

Πίνακας 6.2 Η έξοδος του συστήματος Weka για το αρχείο weather.arff με χρήση ενός ταξινομητή που χρησιμοποιεί νευρωνικά δίκτυα (αλγόριθμος backpropagation).

Στην περίπτωσή μας θα μπορούσαμε να χρησιμοποιήσουμε το Weka προκειμένου να κατηγοριοποιήσουμε τους παίκτες ανάλογα με την απόδοσή τους όσον αφορά στο παιχνίδι. Με τον τρόπο αυτό θα μπορούσαμε να φτιάξουμε κάποια βασικά μοντέλα παικτών αυξανόμενου επιπέδου όσον αφορά την απόδοσή τους στο παιχνίδι που θα τα χρησιμοποιούσαμε για να κρίνουμε την πορεία ήδη εγγεγραμμένων στο σύστημα παικτών ή για να χαρακτηρίσουμε νέους παίκτες.

7. Ανάλυση παιχνιδιού

7.1. Εισαγωγή

Στο κεφάλαιο αυτό θα προσπαθήσουμε να αναλύσουμε το παιχνίδι μας στηριζόμενοι στη θεωρία που αναπτύξαμε στα προηγούμενα κεφάλαια. Όπως έχουμε ήδη αναφέρει στο Κεφάλαιο 1 το παιχνίδι διεξάγεται πάνω σε μια τετραγωνική σκακιέρα διαστάσεων *nxn.* Οι παίκτες έχουν στη διάθεσή τους *β* πιόνια και μία βάση διαστάσεων *αχα* ο καθένας. Στόχος του κάθε παίκτη είναι να καταλάβει την αντίπαλη βάση προστατεύοντας ταυτόχρονα τη δικιά του. Οι κανόνες σχετικά με την κίνηση των πιονιών και επιπλέον πληροφορίες για τα συστατικά στοιχεία του παιχνιδιού περιγράφονται αναλυτικά στο Κεφάλαιο 1.

7.2. Εφαρμογή της Ενισχυτικής Μάθησης στο παιχνίδι μας

Οι επιστήμονες της θεωρίας των παιχνιδιών χρησιμοποιούν την Μηχανική Μάθηση *(Machine Learning)* προκειμένου να φτιάξουν έξυπνα προγράμματα, ικανά να συναγωνίζονται τους ανθρώπους. Το γεγονός αυτό σε συνδυασμό με την επιθυμία πειραματισμού με μεθόδους Μηχανικής Μάθησης αποτέλεσαν τα εναύσματα για τη χρήση μιας πιο συγκεκριμένης μεθόδου Μηχανικής Μάθησης, της Ενισχυτικής Μάθησης *(Reinforcement Learning)*, στο παιχνίδι μας. Τα ερωτήματα που χρειάστηκε να απαντηθούν κατά τη χρήση της Ενισχυτικής Μάθησης είναι τα ακόλουθα:

- Ποια η φύση του παιχνιδιού;
- Τι πληροφορίες θα περιλαμβάνει η κωδικοποίηση του παιχνιδιού;
- Τι αμοιβή *(reward)* πρέπει να αποδίδεται σε κάθε περίπτωση;
- Πως θα γίνεται η επιλογή των κινήσεων;
- Ποια τιμή πρέπει να χρησιμοποιηθεί για την παράμετρο του ρυθμού μείωσης *γ*;
- Πως θα χρησιμοποιηθεί ο αλγόριθμος TD (λ);
- Ποια ίχνη καταλληλότητας πρέπει να χρησιμοποιηθούν;
- Τι πολυπλοκότητα έχει το παιχνίδι;
- Πως θα υπολογίσουμε τη βέλτιστη στρατηγική;

Ακολούθως παραθέτουμε τις δικές μας προσεγγίσεις για κάθε επιμέρους ερώτημα.

Η φύση του παιχνιδιού

Το παιχνίδι μας αποτελεί μια Πεπερασμένη Μαρκοβιανή Διαδικασία Αποφάσεων *(Finite MDP)*, καθώς το σύνολο των καταστάσεων S και το σύνολο των κινήσεων A του παιχνιδιού είναι πεπερασμένο και οι κινήσεις και οι αμοιβές που αποδίδονται εξαρτώνται μόνο από την τρέχουσα κατάσταση του παιχνιδιού. Η προγενέστερη *(a priori)* γνώση του παιχνιδιού συνίσταται μόνο στους κανόνες του παιχνιδιού.

Το σύνολο των καταστάσεων S είναι όλες οι πιθανές διαμορφώσεις της σκακιέρας που μπορούν να συμβούν κατά τη διεξαγωγή του παιχνιδιού. Το σύνολο των κινήσεων A είναι όλες οι δυνατές κινήσεις όπως προκύπτουν από τους κανόνες κίνησης των πιονιών (βλέπε Κεφάλαιο 1).

Κωδικοποίηση παιχνιδιού

Η κωδικοποίηση του παιχνιδιού πρέπει να είναι μικρή και αποτελεσματική. Τυχόν μεγάλη κωδικοποίηση αυξάνει το μέγεθος του χώρου των καταστάσεων και μειώνει το ρυθμό μάθησης. Απ την άλλη τυχόν μικρή κωδικοποίηση εγκυμονεί τον κίνδυνο απώλειας κάποιας πληροφορίας σημαντικής για τη μάθηση.

Η αμοιβή *(reward)*

Το σύστημα αλληλεπιδρά με το περιβάλλον μάθησης μέσω των αμοιβών που λαμβάνει για τις κινήσεις που επιλέγει. Η απόδοση της αμοιβής *(reward)* γίνεται ως εξής:

- Αν η κατάσταση είναι τελική, ο νικητής επιβραβεύεται με *+1* 0 και ο άλλος παίκτης τιμωρείται με *-10*.
- Αν είναι η σειρά του λευκού παίκτη και ο λευκός μπορεί να κερδίσει (το μπορεί να κερδίσει σημαίνει πως κάποιο πιόνι του βρίσκεται δίπλα στη σκακιέρα του μαύρου παίκτη), τότε ο λευκός επιβραβεύεται με +2. Αν μπορεί να κερδίσει ο μαύρος, ο λευκός τιμωρείται με –2, ενώ αν μπορούν να κερδίσουν και οι δύο δίνουμε αμοιβή –1 και στους δύο.
- Αν είναι η σειρά του μαύρου παίκτη και ο μαύρος μπορεί να κερδίσει (το μπορεί να κερδίσει σημαίνει πως κάποιο πιόνι του βρίσκεται δίπλα στη σκακιέρα του λευκού παίκτη), τότε ο μαύρος επιβραβεύεται με +2. Αν μπορεί να κερδίσει ο λευκός, ο μαύρος τιμωρείται με –2, ενώ αν μπορούν να κερδίσουν και οι δύο δίνουμε αμοιβή –1 και στους δύο.
- Για τις υπόλοιπες περιπτώσεις η αμοιβήυπολογίζεται ως εξής:

  Βρίσκουμε κάθε φορά τη διαφορά των πιονιών των δύο αντιπάλων (οι παίκτες μπορεί να έχασαν κάποια πιόνια κατά τη διεξαγωγή του παιχνιδιού) και την πολλαπλασιάζουμε με τον παράγοντα *1/(αρχικό πλήθος πιονιών)*. Αν π.χ. αρχικά ο κάθε παίκτης είχε 10 πιόνια και η διαφορά των πιονιών είναι -6 για το λευκό παίκτη, δηλαδή ο λευκός έχει 6 λιγότερα πιόνια από το μαύρο, τότε θα δοθεί στο λευκό αμοιβή -0,6.

Επιλογή των κινήσεων

Το παιχνίδι θα πρέπει να χρησιμοποιήσει την αμοιβή *(reward)* για την εκπαίδευσή του. Πιο συγκεκριμένα θα πρέπει να αξιολογήσει τις κινήσεις του βάσει της αμοιβής που του επέφεραν και να μάθει να επιλέγει εκείνες τις κινήσεις που θα του επιφέρουν τη μεγαλύτερη αμοιβή.

Ωστόσο το σύστημα δε θα πρέπει να επιλέγει πάντα τις κινήσεις που έχουν τη μεγαλύτερη αμοιβή γιατί έτσι δεν εξερευνεί νέες κινήσεις που πιθανόν να έχουν μεγαλύτερη αμοιβή.

Στην περίπτωση μας η επιλογή των κινήσεων δεν καθορίζεται πάντα με βάση τη μεγαλύτερη αναμενόμενη αμοιβή. Το σύστημα επιλέγει κινήσεις χρησιμοποιώντας μια άπληστη στρατηγική *(ε- greedy policy)* με *ε=0.9*. Αυτό σημαίνει πως στο 90% των περιπτώσεων το σύστημα επιλέγει εκείνες τις κινήσεις που έχουν τη μεγαλύτερη αναμενόμενη αμοιβή, δηλαδή εκμεταλλεύεται την γνώση που ήδη κατέχει (*exploitation)* και στο υπόλοιπο 10% των περιπτώσεων το σύστημα επιλέγει τυχαίες κινήσεις, δηλαδή ρισκάρει εξερευνώντας νέες κινήσεις *(exploration)*.

Η παράμετρος του ρυθμού μείωσης

Όσον αφορά στην παράμετρο του ρυθμού μείωσης γ *(discount rate parameter)*, η οποία καθορίζει την αξία των μελλοντικών αμοιβών, χρησιμοποιήθηκε την τιμή *=0.95* προκειμένου το σύστημα να λαμβάνει σοβαρά υπόψη του τις μελλοντικές αμοιβές, προσδοκώντας έτσι το σύστημα να είναι πιο δίκαιο και να

υιοθετήσει μια μακροπρόθεσμη στρατηγική. Η τιμή αυτή δικαιολογείται και από το γεγονός ότι το παιχνίδι μας είναι πεπερασμένου διακριτού χρόνου.

Ο Αλγόριθμος TD (λ)

Χρησιμοποιήθηκε ο αλγόριθμος μάθησης χρονικών διαφορών *(temporal difference learning algorithm)* TD (λ) και πιο συγκεκριμένα η on-line έκδοση του όπου η ανανέωση λαμβάνει χώρα σε κάθε βήμα. Για τον παράγοντα παράληψης *(forgetting factor) λ* που καθορίζει μέχρι ποιο σημείο ισχύει η ανάθεση πίστωσης *(credit assignment)* χρησιμοποιήθηκε η τιμή *λ=0.5,* προκειμένου το λάθος που μπορεί να συμβεί μια δεδομένη χρονική να μεταφερθεί μόνο στις τελευταίες 6-7 κινήσεις του παιχνιδιού.

Ίχνη καταλληλότητας

Χρησιμοποιήθηκαν τα ίχνη αντικατάστασης *(replacing traces)* και όχι τα ίχνη συσσώρευσης *(accumulating traces)* καθώς τα τελευταία έχουν κάποια γνωστά μειονεκτήματα με πιο σημαντικό το πρόβλημα ότι μια επαναλαμβανόμενη λάθος κίνηση εμποδίζει τη μάθηση (το ίχνος της κακής κίνησης αυξάνεται). Τα ίχνη καταλληλότητας ανανεώνονται σε κάθε βήμα.

Πολυπλοκότητα παιχνιδιού

Η πολυπλοκότητα του παιχνιδιού εξαρτάται από τον τρόπο με τον οποίο τα πιόνια των δύο αντιπάλων είναι τοποθετημένα στη σκακιέρα. Βασικοί παράμετροι είναι η διάσταση της σκακιέρας *n,* η διάσταση της βάσης *α* και το πλήθος των πιονιών *β.* Ένα άνω όριο για τις πιθανές καταστάσεις του παιχνιδιού είναι:

$$\sum_{i=0}^{\beta}\sum_{j=0}^{\beta}\binom{n^2}{i+j}\binom{2a^2}{i}\binom{\beta+1-i}{\beta+1-j}$$

Υπολογισμός βέλτιστης στρατηγικής

Για μεγάλες τιμές των παραμέτρων *n,α, β* η πολυπλοκότητα του παιχνιδιού αυξάνεται δραματικά, με αποτέλεσμα η χρήση ενός πίνακα αντιστοίχησης να μην είναι αποδοτική. Προκύπτει λοιπόν, η ανάγκη εύρεσης κάποιου τρόπου γενίκευσης. Στην περίπτωσή μας χρησιμοποιήθηκαν τα νευρωνικά δίκτυα για τη γενίκευση.

7.3. Εφαρμογή των νευρωνικών δικτύων στο παιχνίδι μας

Εξηγήσαμε μόλις πριν τους λόγους για τους οποίους χρησιμοποιήθηκαν τα νευρωνικά δίκτυα στο παιχνίδι μας. Τα ερωτήματα που χρειάστηκε να απαντηθούν όσον αφορά στα νευρωνικά δίκτυα είναι τα ακόλουθα:

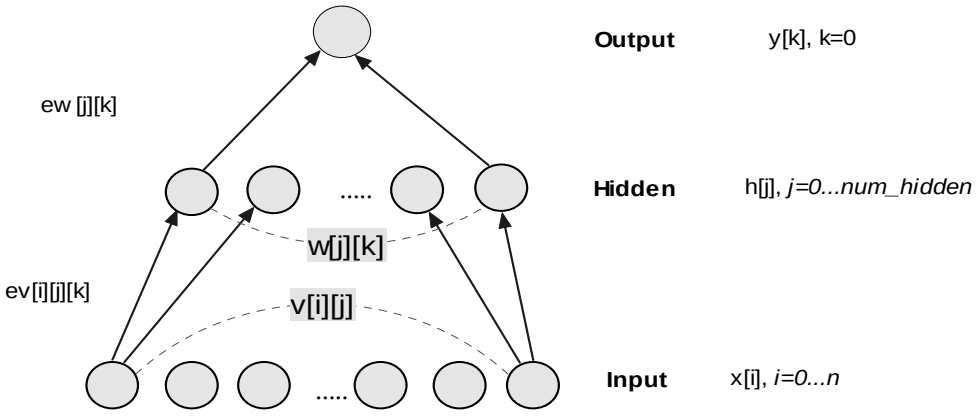
- Ποια θα είναι η αρχιτεκτονική του δικτύου;
- Ποια δεδομένα θα αποτελούν την είσοδο του δικτύου;
- Πως θα υπολογισθεί η αξία της εισόδου στην έξοδο;
- Πως θα υπολογισθεί το λάθος στους νευρώνες εξόδου;
- Πως θα ανανεώνονται τα βάρη του δικτύου;
- Τι τιμές πρέπει να έχουν οι σταθερές παράμετροι του δικτύου;
- Πως θα επιτευχθεί η μάθηση;

Στην συνέχεια παραθέτουμε τις δικές μας προσεγγίσεις για κάθε επιμέρους ερώτημα.

Η αρχιτεκτονική του δικτύου

Χρησιμοποιήθηκαν δύο νευρωνικά δίκτυα, ένα για κάθε παίκτη, εξαιτίας του ότι ο χώρος των καταστάσεων του ενός παίκτη δεν έχει κοινά στοιχεία με το χώρο των καταστάσεων του άλλου παίκτη. Έτσι κάθε παίκτης έχει να μάθει ένα μοναδικό χώρο καταστάσεων και μια συγκεκριμένη διαρρύθμιση της σκακιέρας για τον ένα παίκτη δε θα εμφανιστεί ποτέ στον άλλο παίκτη. Ο αλγόριθμος που χρησιμοποιήθηκε για την εκπαίδευση είναι ο «vanilla» backpropagation.

Η αρχιτεκτονική καθενός εκ των δύο νευρωνικών δικτύων φαίνεται στο ακόλουθο σχήμα (Σχήμα 7.1):



Σχήμα 7.1 Η αρχιτεκτονική του νευρωνικού δικτύου για κάθε παίκτη

Το κάθε νευρωνικό αποτελείται από τρία επίπεδα νευρώνων:

- το επίπεδο εισόδου *(input layer)* που περιέχει $2*(n^2-2*a^2+5)$ νευρώνες
- το κρυμμένο επίπεδο *(hidden layer)* που περιέχει n^2-2*a^2+5 νευρώνες
- το επίπεδο εξόδου *(output layer)* που περιέχει *1* νευρώνα.

όπου: *n:* η διάσταση της σκακιέρας, *a:* η διάσταση της βάσης.

Η είσοδος του δικτύου

Ως είσοδος στο νευρωνικό χρησιμοποιήθηκε μια αναπαράσταση της σκακιέρας μέσω των πιονιών των δύο αντιπάλων. Προκειμένου όμως να μπορέσει να χρησιμοποιηθεί η αναπαράσταση αυτή ως είσοδος το νευρωνικό έπρεπε να μετατραπεί σε δυαδική μορφή. Η μετατροπή έγινε ως εξής: Για καθένα από τα $DIMBOARD^2-2*DIMBASE^2$ τετράγωνα της σκακιέρας υπάρχουν 2 τιμές, μία για κάθε παίκτη, που είτε είναι και οι δύο μηδέν είτε κάποια από τις δύο (ανάλογα με τον παίκτη) είναι ένα – το ένα σημαίνει πως υπάρχει πιόνι στο συγκεκριμένο τετραγωνάκι. Μετρήσαμε επίσης πόσα πιόνια είναι ακόμη μέσα στη βάση και χρησιμοποιήσαμε 4 νευρώνες για κάθε παίκτη για να δηλώσουμε το ποσοστό των πιονιών που είναι ακόμη μέσα στη βάση. Υπάρχουν 2 ακόμη νευρώνες, ένας για κάθε παίκτη που γίνονται ένα όταν αυτός είναι ο νικητής. Συνολικά δηλαδή χρησιμοποιήθηκαν 2($DIMBOARD^2-2*DIMBASE^2+5$) νευρώνες εισόδου, ($DIMBOARD^2-2*DIMBASE^2+5$) κρυμμένοι νευρώνες και 1 νευρώνας εξόδου.

Υπολογισμός της αξίας της εισόδου

Δοθείσας μιας τέτοιας αναπαράστασης των πιονιών, το νευρωνικό υπολογίζει την αξία αυτής της εισόδου και το αποτέλεσμα φαίνεται στην έξοδο. Ο υπολογισμός της αξίας μιας εισόδου πραγματοποιείται με τη

$$h(j) = \frac{1}{1 + e^{-\sum_I w_{ij} \phi_j}}$$

βοήθεια μιας σιγμοειδούς συνάρτησης της μορφής

που παίρνει τιμές στο διάστημα [0-1] σε δύο βήματα:

1. Πρώτα υπολογίζεται η αξία των νευρώνων του κρυμμένου επιπέδου μέσω της συνάρτησης:

$$hiddenNode[j] = \frac{1}{1 + e^{-\{\sum_{i=0}^{inputNodes} inputNode[i] * v[i][j]\}}}$$

όπου

hiddenNode[j]: η τιμή κάποιου κρυμμένου κόμβου *j*

inputNode[i]: η τιμή κάποιου κόμβου εισόδου

v[i][j]: το βάρος της σύνδεσης μεταξύ του επιπέδου εισόδου και του κρυμμένου επιπέδου (βάρος πρώτου επιπέδου)

2. Στη συνέχεια υπολογίζεται η αξία των νευρώνων του επιπέδου εξόδου μέσω της συνάρτησης:

$$outputNode[k] = \frac{1}{1 + e^{-\{\sum_{j=0}^{hiddenNode} s\, hiddenNode[j] * w[j][k]\}}}$$

όπου

outputNode[k]: η τιμή κάποιου κόμβου εξόδου *k*

w[j][k]: το βάρος της σύνδεσης μεταξύ του κρυμμένου επιπέδου και του επιπέδου εξόδου (βάρος δευτέρου επιπέδου)

Τα βήματα 1 και 2 είναι γνωστά ως η φάση της ενεργοποίησης της έμπροσθεν διασποράς *(activation forward propagation phase).*

Υπολογισμός του λάθους του νευρώνα εξόδου

Στη συνέχεια υπολογίζεται το λάθος στους νευρώνες εξόδου μέσω της συνάρτησης:

$$\text{error}[k] = reward[k] + \gamma * outputNode[k] - oldoutputNode[k]$$

όπου

error[k]: το λάθος για τον κόμβο k του επιπέδου εξόδου

reward[k]: η αμοιβή *(reward)* που λαμβάνει ο agent για την απόφασή του

γ: ο ρυθμός μείωσης του λάθους

oldOutputNode[k]: χρησιμοποιείται για την ανανέωση των βαρών

Ανανέωση των βαρών του δικτύου

Ακολουθεί η ανανέωση των βαρών των συνδέσεων του νευρωνικού. Για τα μεν βάρη του δευτέρου επιπέδου η ανανέωση πραγματοποιείται μέσω της συνάρτησης:

$$w[j][k] = w[j][k] + \{\beta * error[k] * ew[j][k]\}$$

όπου

β: ο ρυθμός μάθησης του δευτέρου επιπέδου που δίνεται από τη σχέση:

$$\beta = \frac{1}{hiddenNode_s}$$

ew[j][k]: το eligibility trace της εξόδου

Για τα δε βάρη του πρώτου επιπέδου η ανανέωση πραγματοποιείται μέσω της συνάρτησης:

$$v[i][j] = v[i][j] + \{\alpha * error[k] * ev[i][j][k]\}$$

όπου

α: ο ρυθμός μάθησης του πρώτου επιπέδου που δίνεται από τη σχέση:

$$\alpha = \frac{1}{inputNodes}$$

ev[l][j][k]: το eligibility trace του κρυμμένου επιπέδου

Το βήμα αυτό είναι γνωστό ως η φάση της προς τα πίσω διασποράς του λάθους *(error backward propagation phase).*

Διαδικασία μάθησης

Πριν αρχίσει το νευρωνικό να μαθαίνει τα βάρη αρχικοποιούνται με τυχαίες μικρές τιμές. Η μάθηση συντελείται ως εξής: Το νευρωνικό ενεργοποιείται με κάποια είσοδο και υπεισέρχεται στη φάση της ενεργοποίησης της έμπροσθεν διασποράς *(activation forward propagation phase)*: υπολογίζονται οι τιμές των κρυμμένων επιπέδων και των επιπέδων εξόδου μέσω μιας σιγμοειδούς συνάρτησης ενεργοποίησης *(sigmoid activation function).*

Μετά τη φάση της ενεργοποίησης της έμπροσθεν διασποράς *(activation forward propagation phase)* ακολουθεί η φάση της προς τα πίσω διασποράς του λάθους *(error backward propagation phase)* όπου υπολογίζεται το λάθος για το επίπεδο εξόδου. Στη συνέχεια τα βάρη των συνδέσεων του νευρωνικού (πρώτο και δεύτερο επίπεδο) ανανεώνονται.

7.4. Μοντελοποίηση των παικτών

Κάθε παίκτης αποτελεί μοναδική οντότητα για το σύστημα (πιστοποιείται μέσω του αναγνωριστικού του *(login)*) και έχει το δικό του μοντέλο στο οποίο αποθηκεύονται πληροφορίες όπως:

- Το πλήθος των χαμένων παιχνιδιών.
- Το πλήθος των κερδισμένων παιχνιδιών.
- Το πλήθος των κινήσεων για τα κερδισμένα παιχνίδια
- Το πλήθος των κινήσεων για τα χαμένα παιχνίδια

- Ο μέσος όρος της αξίας των κινήσεων του παίκτη ανά παιχνίδι
- Η μέση τετραγωνική διαφορά από τις καλύτερες κάθε φορά προτεινόμενες κινήσεις του νευρωνικού.
- Η μέση τετραγωνική διαφορά από τις χειρότερες κάθε φορά προτεινόμενες κινήσεις του νευρωνικού.

Κάθε παίκτης κατατάσσεται σε ένα μοντέλο ανάλογα με το πλήθος των παιχνιδιών που έχει παίξει και τα αποτελέσματα αυτών των παιχνιδιών. Ο καθορισμός των μοντέλων θα θέλαμε να γίνει με βάση τα αποτελέσματα κάποιου συστήματος κατηγοριοποίησης *(classification system)* π. χ. του Weka. Για λόγους που θα αναφέρουμε εκτενέστερα παρακάτω κάτι τέτοιο δεν ήταν εφικτό με αποτέλεσμα να υποθέσουμε τέσσερα πιθανά μοντέλα, τα ακόλουθα:

Άγνωστος *(Unknown)*: ένας παίκτης θεωρείται αγνώστου μοντέλου αν το πλήθος των συνολικών παιχνιδιών που έχει παίξει είναι μικρότερο του 4.

Αρχάριος (Beginner*)*: ένας παίκτης θεωρείται αρχάριος αν το ποσοστό των κερδισμένων παιχνιδιών στο σύνολο των παιχνιδιών που έχει παίξει ο παίκτης είναι μικρότερο ή ίσο του 40%.

Προχωρημένος *(Advanced)*: ένας παίκτης θεωρείται προχωρημένος αν το ποσοστό των κερδισμένων παιχνιδιών στο σύνολο των παιχνιδιών που έχει παίξει ο παίκτης είναι μεγαλύτερο του 40% και μικρότερο ή ίσο του 80%.

Ειδικός *(Expert)*: ένας παίκτης θεωρείται ειδικός αν το ποσοστό των κερδισμένων παιχνιδιών στο σύνολο των παιχνιδιών που έχει παίξει ο παίκτης είναι μεγαλύτερο του 80%.

Μετά από κάθε παιχνίδι το μοντέλο ενημερώνεται με τα νέα δεδομένα.

7.5. Εκπαίδευση - Πειράματα

Στα πλαίσια της επιστημονικής δημοσίευσης [Kalles & Kanellopoulos 2001] υλοποιήθηκε μια σειρά πειραμάτων για την αξιολόγηση του παιχνιδιού. Τα αποτελέσματα αυτών των πειραμάτων συνοψίζονται στον παρακάτω πίνακα:

Πλήθος παιχνιδιών*10,000	**Μέσος όρος κινήσεων**	**Ποσοστό επιτυχίας για τον λευκό παίκτη**	**Ποσοστό επιτυχίας για τον μαύρο παίκτη**
0 – 1	179.94	47.83%	52.17%
1 – 2	182.88	47.72%	52.28%
2 – 3	187.87	50.30%	49.70%
3 – 4	186.88	49.98%	50.02%
4 – 5	184.90	50.03%	49.97%
5 – 6	183.53	49.00%	51.00%
6 – 7	184.76	49.79%	50.21%
7 –7.4	186.37	49.85%	50.15%

Πίνακας 7.1 Αποτελέσματα των πειραμάτων [Kalles & Kanellopoulos 2001]

Από τα αποτελέσματα φαίνεται πως καθώς οι διαστάσεις της σκακιέρας αυξάνονται, ο μέσος όρος των κινήσεων αυξάνεται σημαντικά. Ωστόσο τα αποτελέσματα αυτά προέρχονται από παιχνίδια του υπολογιστή με τον εαυτό του *(self-playing games)* και δεν μπορεί κανείς να βγάλει συμπεράσματα για την περίπτωση που παίζει κάποιος άνθρωπος με τον υπολογιστή.

Κατά την διάρκεια της διπλωματικής όμως υλοποιήθηκε η δυνατότητα παιχνιδιού μεταξύ ανθρώπου και υπολογιστή και τα αποτελέσματα έδειξαν πως ο υπολογιστής δεν είχε εκπαιδευτεί καλά καθώς στην συντριπτική πλειοψηφία των περιπτώσεων ο άνθρωπος κέρδιζε ακόμα και αν ακολουθούσε τυχαίες κινήσεις. Για το λόγο αυτό επανεξετάσαμε την εφαρμογή της Ενισχυτικής Μάθησης και των Νευρωνικών Δικτύων στο παιχνίδι μας και τροποποιήσαμε σε κάποια σημεία τη λογική του προγράμματος. Τρέξαμε το νέο πρόγραμμα για ένα σύνολο 50000 πειραμάτων, Τα αποτελέσματα των πειραμάτων δίνουν ποσοστό επιτυχίας 56,7% στο λευκό παίκτη και 43,3% στο μαύρο παίκτη.

Αξιολόγηση πειραμάτων

Τα αποτελέσματα των νέων πειραμάτων είναι λογικά δεδομένου ότι ο λευκός παίκτης ξεκινάει πρώτος και συνεπώς το ποσοστό επιτυχίας του είναι μεγαλύτερο από το ποσοστό επιτυχίας του μαύρου παίκτη. Και πάλι όμως παίζοντας με άνθρωπο αντίπαλο ο υπολογιστής χάνει σχεδόν πάντα, αν και πλέον οι κινήσεις του δεν είναι τόσο τυχαίες. Ο υπολογιστής φαίνεται να έχει μάθει πως είναι επικίνδυνο να πλησιάζουν στη βάση του τα πιόνια του αντιπάλου και για το λόγο αυτό βγάζει πολλά πιόνια γύρω από τη βάση και «τρώει» τα πιόνια του αντιπάλου στις περιπτώσεις που αυτό είναι εφικτό. Ωστόσο ακόμη η ταχύτητα με την οποία ο υπολογιστής προσεγγίζει τη βάση του αντιπάλου είναι μικρή πράγμα που σημαίνει πως δεν έχει μάθει ότι απώτερος σκοπός του είναι η νίκη επί του αντιπάλου. Ευελπιστούμε γι'αυτό να ευθύνεται η μικρή ως προς το πλήθος των πειραμάτων εκπαίδευση.

7.6. Αρχιτεκτονική παιχνιδιού

Η υλοποίηση του παιχνιδιού έγινε σε γλώσσα προγραμματισμού JAVA και συνίσταται σε 16 κλάσεις – αρχεία, τα ακόλουθα:

1. Common

Περιλαμβάνει κάποιες καθολικές μεταβλητές που είναι προσπελάσιμες από όλα τα υπόλοιπα αρχεία του συστήματος.

2. Class1

Είναι η κύρια κλάση του συστήματος μέσω της οποίας γίνεται και η πιστοποίηση του παίκτη.

3. GameBoard

Δημιουργία της σκακιέρας του παιχνιδιού και των επιμέρους τετραγώνων της.

4. History

Καταγραφή των αποτελεσμάτων των παιχνιδιών και των κινήσεων των παικτών.

5. NeuralNet

Δημιουργία – διαχείριση του νευρωνικού δικτύου

6. Pawn

Δημιουργία – διαχείριση πιονιών

7. Player

Δημιουργία – διαχείριση παίκτη

8. Position

Δημιουργία – διαχείριση της διαμόρφωσης της σκακιέρας.

9. Square

Δημιουργία – διαχείριση των τετραγώνων της σκακιέρας.

10. Spiel

Δημιουργία – διαχείριση ενός παιχνιδιού.

11. Model

Δημιουργία – διαχείριση του μοντέλου του παίκτη.

12. PlotXY

Δημιουργία γραφικής αναπαράστασης σε διδιάστατο χώρο.

13. VisualBoard

Δημιουργία – Οπτικοποίηση της σκακιέρας του παιχνιδιού.

14. VisualBoardFrame

Δημιουργία – Οπτικοποίηση του κεντρικού παραθύρου του παιχνιδιού.

15. infoWindow

Δημιουργία ενός παραθύρου με δυνατότητα να ανοίγει και νέα παράθυρα.

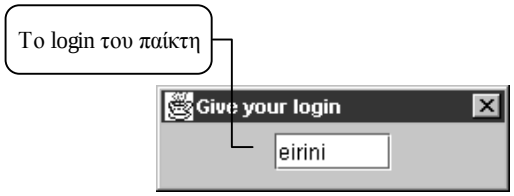
16. SimpleWindow

Δημιουργία ενός απλού παραθύρου.

Οι κλάσεις 1-10 είχαν υλοποιηθεί αρχικά από τους [Kalles & Kanellopoulos 2001]. Στα πλαίσια της διπλωματικής οι παραπάνω κλάσεις τροποποιήθηκαν και προστέθηκαν νέες: η κλάση Model που υλοποιεί τη μοντελοποίηση του παίκτη και οι κλάσεις PlotXY, VisualBoard, VisualBoardFrame, infoWindow, SimpleWindow που υλοποιούν το γραφικό περιβάλλον του παιχνιδιού.

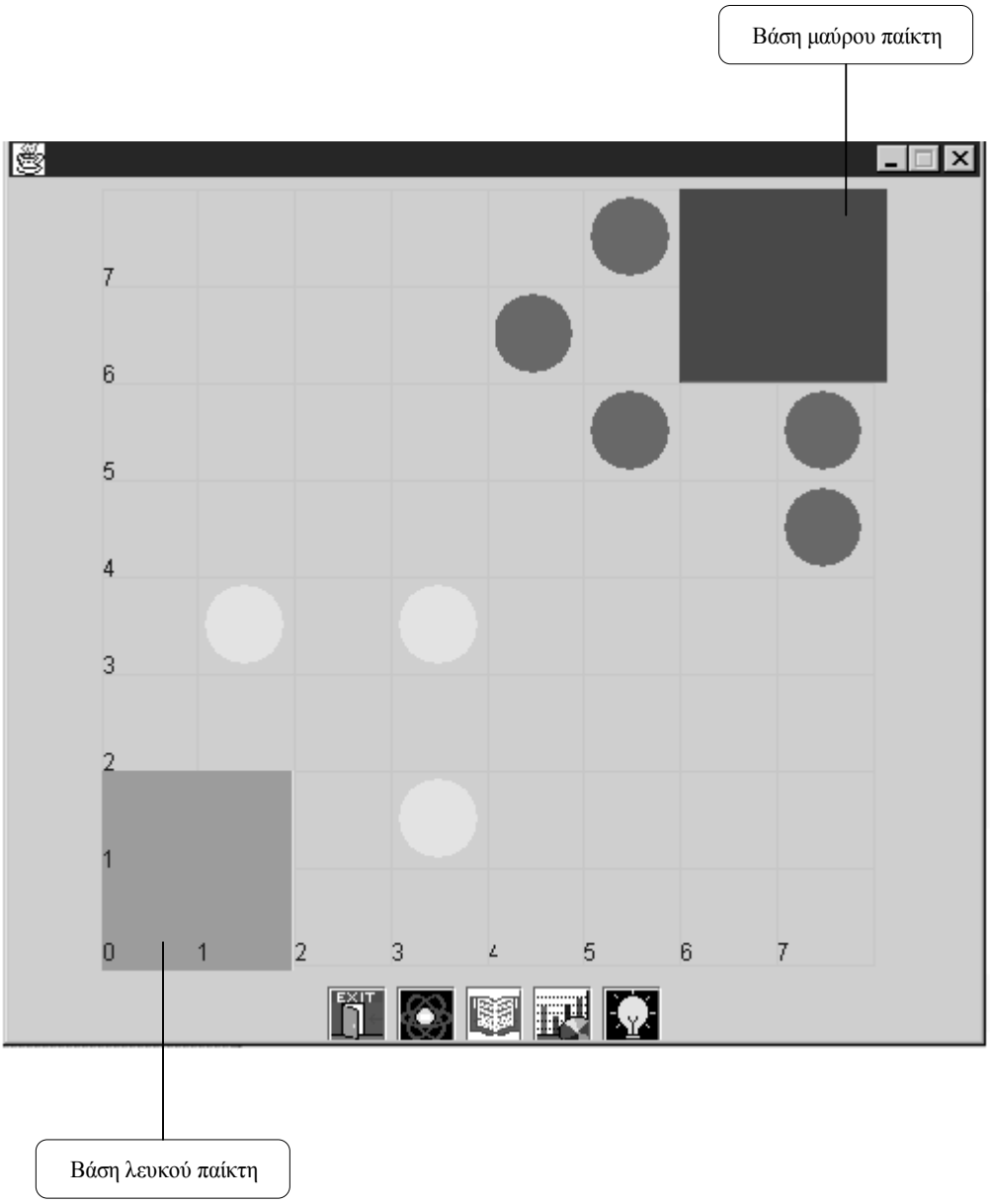
7.7. Το περιβάλλον αλληλεπίδρασης με το παιχνίδι

Ξεκινώντας το παιχνίδι εμφανίζεται η φόρμα πιστοποίησης παίκτη (Σχήμα 7.2). Η πιστοποίηση γίνεται μέσω του login του παίκτη και είναι απαραίτητη προκειμένου να έχουμε στοιχεία για την σταδιακή του πρόοδο. Τα στοιχεία που αφορούν το μοντέλο του παίκτη αποθηκεύονται σε ένα αρχείο με το όνομα του παίκτη και τη σκακιέρα στην οποία παίζει, π.χ. eirini_Model_{8210}.txt όπου το eirini είναι το login του παίκτη και το 8210 είναι η οι διαστάσεις της σκακιέρας δηλαδή διάσταση σκακιέρας, διάσταση βάσης και πλήθος πιονιών. Αν ο παίκτης έχει ξαναπαίξει, τα στοιχεία του φορτώνονται απ' το αντίστοιχο αρχείο, ενώ αν είναι καινούριος αυτόματα δημιουργείται ένα αρχείο για την αποθήκευση των στοιχείων του.

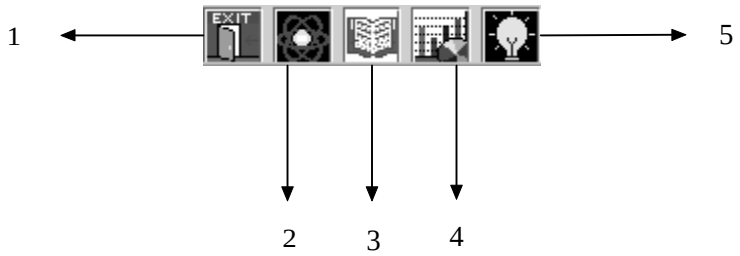


Σχήμα 7.2 Φόρμα πιστοποίησης χρήστη

Μετά την επιτυχή πιστοποίηση παίκτη προβάλλει το κεντρικό παράθυρο του παιχνιδιού (Σχήμα 7.3) που αποτελείται από τη σκακιέρα (η default σκακιέρα είναι διαστάσεων 8x2x10) και 5 buttons που επιτελούν διαφορετικές λειτουργίες:

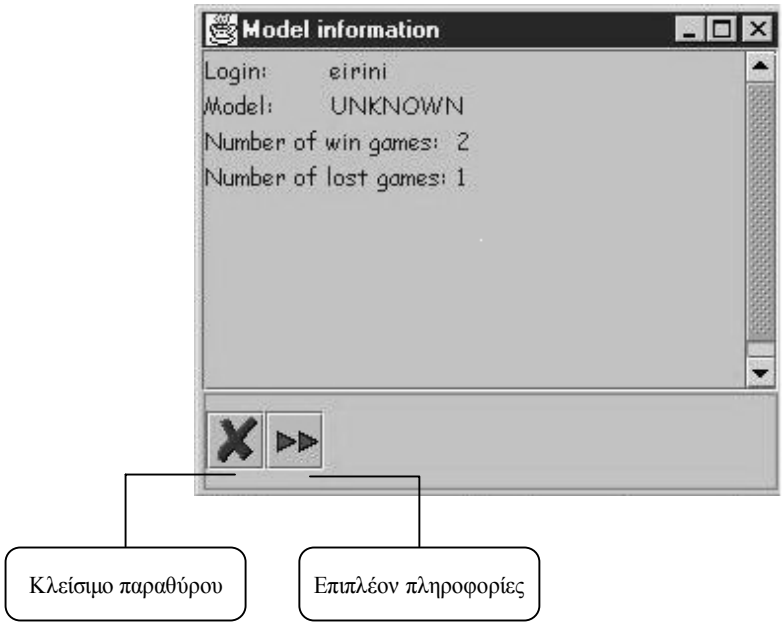


Σχήμα 7.3 Το κεντρικό παράθυρο του παιχνιδιού



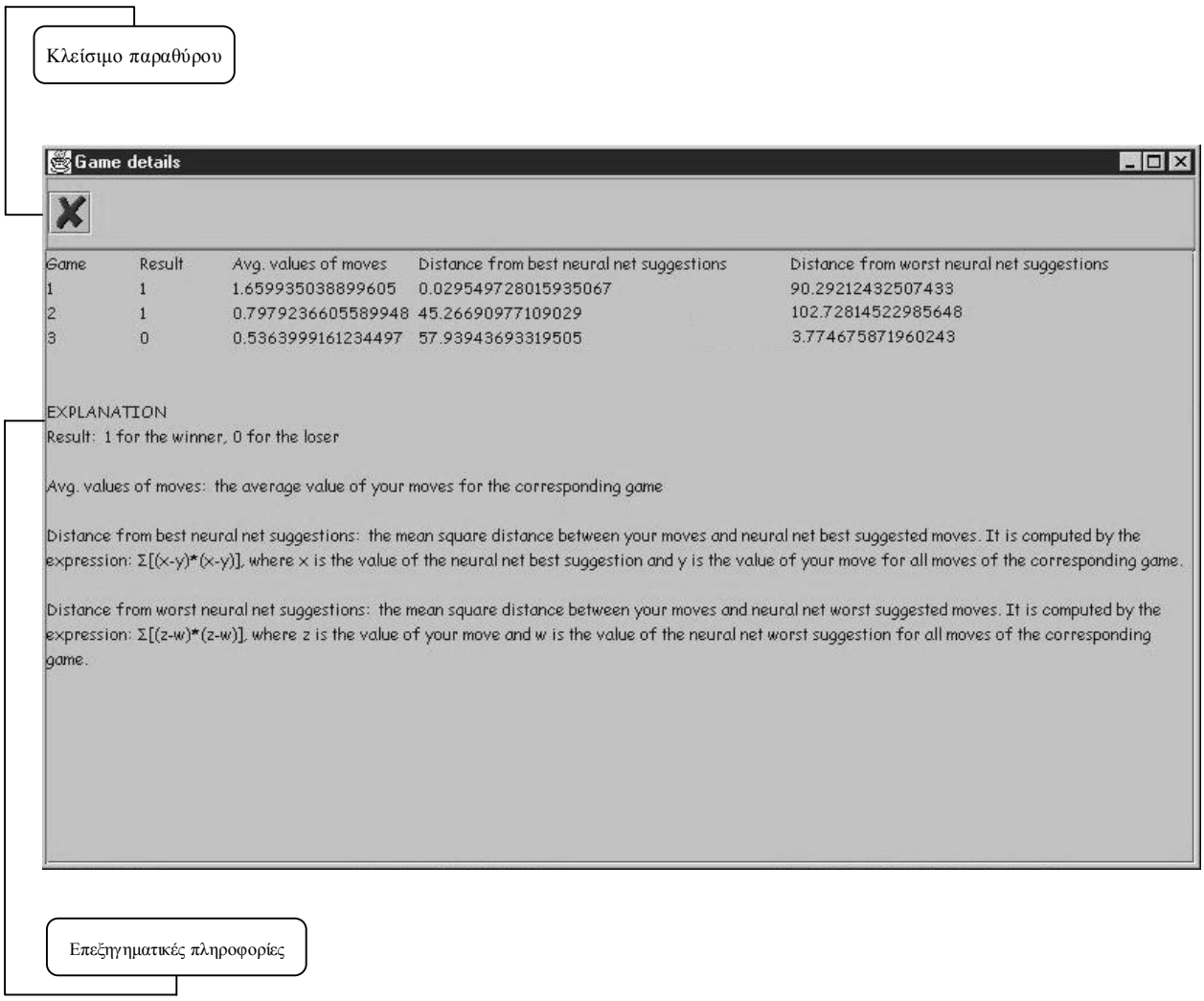
Σχήμα 7.4 Τα buttons του παιχνιδιού

1. Πατώντας το button 1 το τρέχον παιχνίδι τερματίζεται και βγαίνουμε από το πρόγραμμα.
2. Πατώντας το button 2 ανοίγει ένα παράθυρο με πληροφορίες για το μοντέλο του παίκτη (Σχήμα 7.5)



Σχήμα 7.5 Το παράθυρο με τις πληροφορίες για το μοντέλο του παίκτη

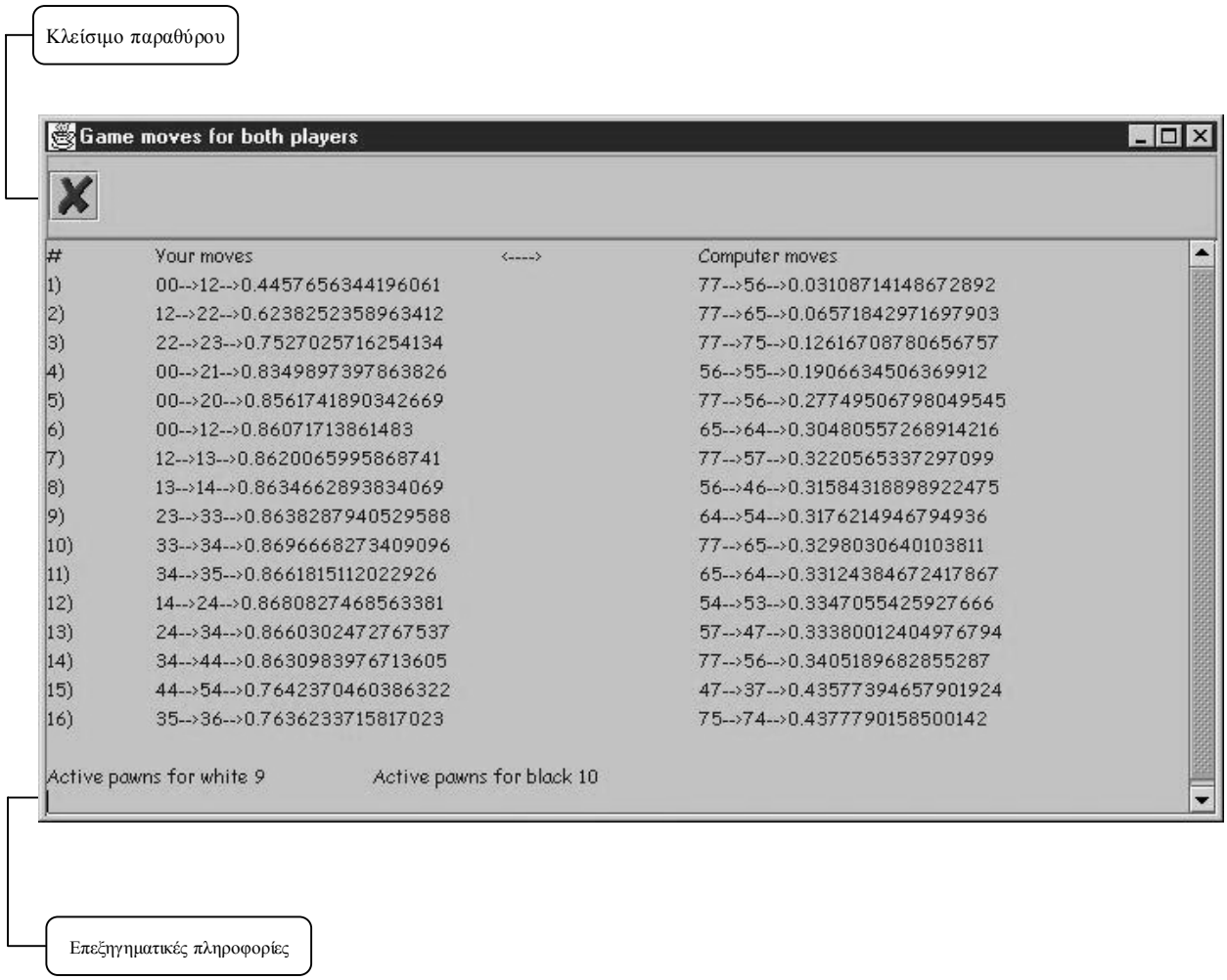
- Πατώντας το πρώτο button κλείνει το παράθυρο
- Πατώντας το δεύτερο button εμφανίζεται ένα νέο παράθυρο (Σχήμα 7.6) με περισσότερες πληροφορίες για τον τρόπο που παίζει ο παίκτης σε σχέση με τις κινήσεις που του προτείνει το νευρωνικό (επισημαίνουμε και πάλι πως θεωρούμε ότι η γνώση του νευρωνικού είναι σωστή και συνεπώς στόχος του παίκτη είναι να βρίσκεται όσο το δυνατόν πιο κοντά στις καλύτερες κινήσεις του νευρωνικού).



Σχήμα 7.6 Επιπλέον πληροφορίες για κάθε παιχνίδι του παίκτη

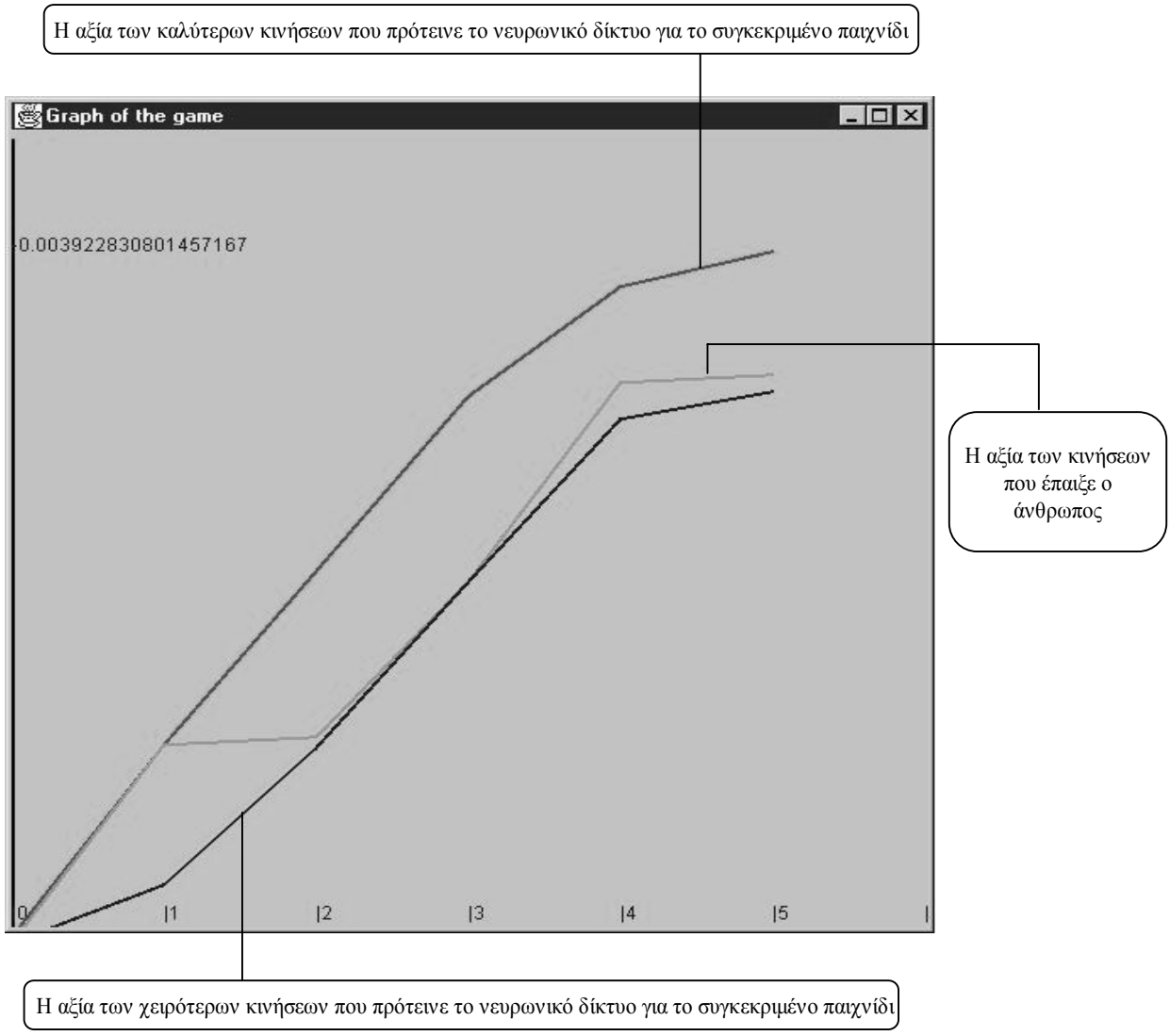
Στο παράθυρο με τις περισσότερες πληροφορίες (Σχήμα 7.6) εμφανίζονται πληροφορίες για όλα τα παιχνίδια του παίκτη. Πιο συγκεκριμένα για κάθε παιχνίδι ο παίκτης μπορεί να δει τη μέση βαθμολογία των κινήσεών του καθώς επίσης και τη μέση βαθμολογία των καλύτερων και χειρότερων κινήσεων που του πρότεινε το νευρωνικό για το συγκεκριμένο παιχνίδι. Με τον τρόπο αυτό, ο παίκτης γνωρίζει ανά πάσα στιγμή τη συγκριτική του θέση σε σχέση με το νευρωνικό.

3. Πατώντας το button 3 ανοίγει ένα παράθυρο (Σχήμα 7.7) με το ιστορικό των κινήσεων του παιχνιδιού. Ο παίκτης μπορεί να βλέπει καθ' όλη τη διάρκεια του παιχνιδιού τόσο τις δικές του κινήσεις και την αξία τους όσο και τις κινήσεις του αντιπάλου και αντίστοιχή τους αξία. Επίσης μπορεί να βλέπει ανά πάσα στιγμή το πλήθος των πιονιών του κάθε παίκτη.



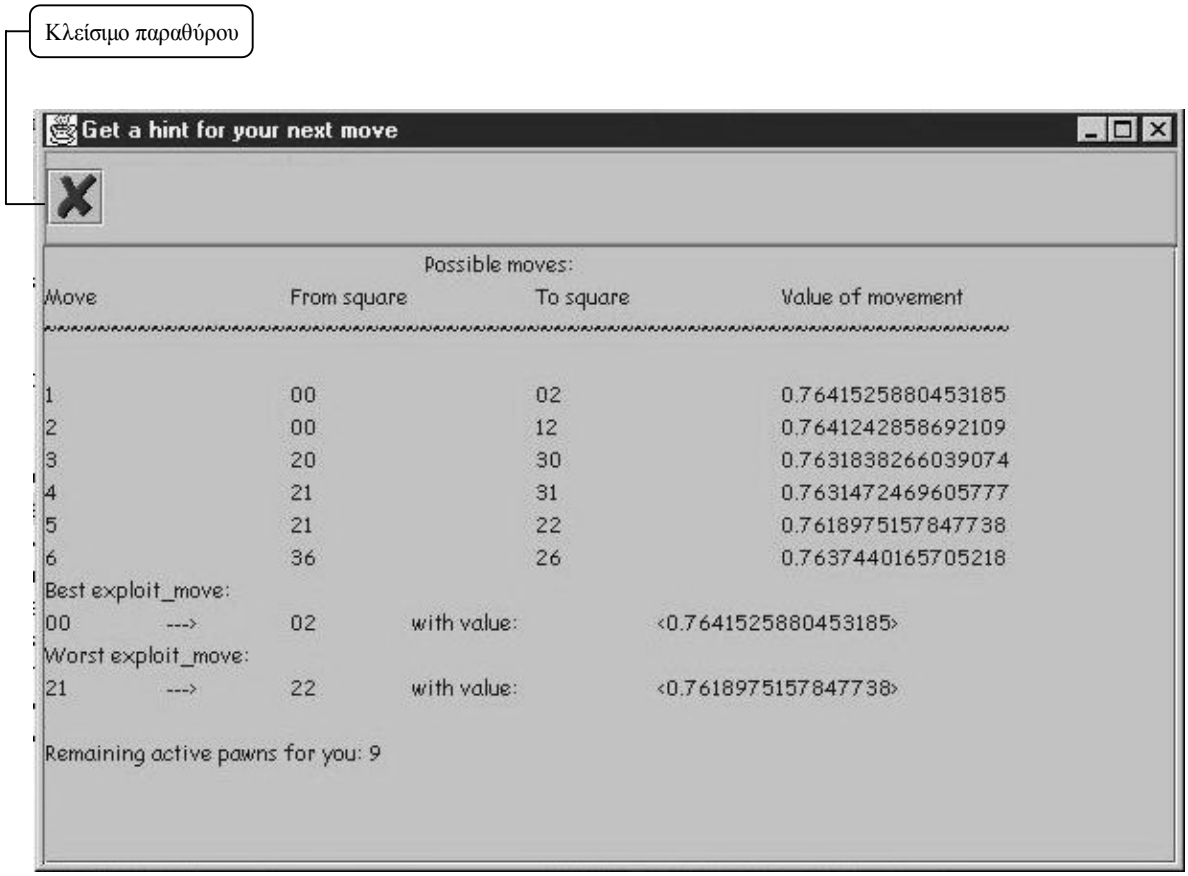
Σχήμα 7.7 Το ιστορικό των κινήσεων του παιχνιδιού

4. Πατώντας το button 4 ανοίγει ένα παράθυρο (Σχήμα 7.8) με τη γραφική παράσταση της αξίας των κινήσεων του παίκτη σε σχέση με την αξία των καλύτερων και χειρότερων κινήσεων που του πρότεινε το νευρωνικό. Με τον τρόπο αυτό ο παίκτης βλέπει με γραφικό τρόπο τη συγκριτική του θέση σε σχέση με τις προτάσεις του νευρωνικού και αποκτά με εύκολο και γρήγορο τρόπο μία συνοπτική εικόνα της κατάστασής του όσον αφορά την απόδοσή του στο παιχνίδι (και πάλι επισημαίνουμε πως έχουμε υποθέσει ότι η γνώση του νευρωνικού είναι σωστή).



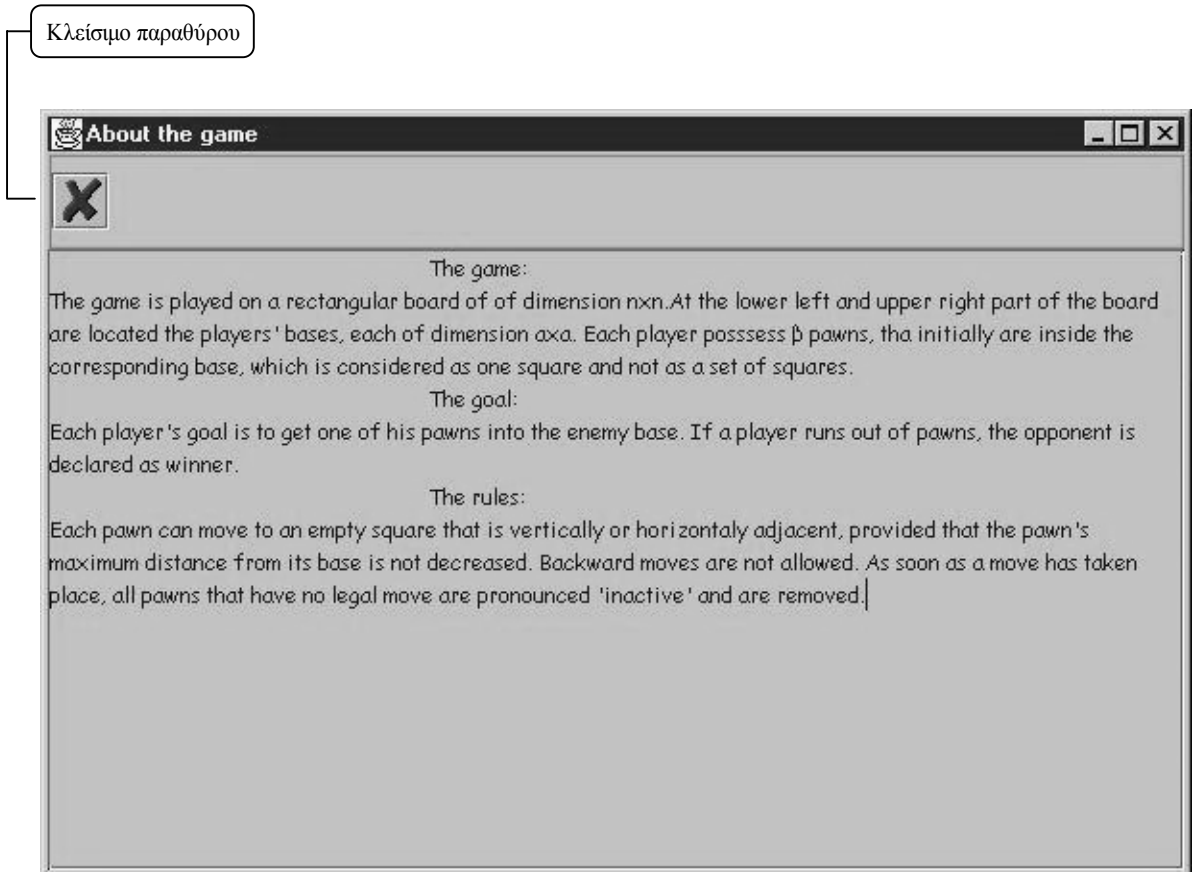
Σχήμα 7.8 Γραφική αναπαράσταση της αξίας των κινήσεων του παίκτη σε σχέση με τι; καλύτερες και χειρότερες κινήσεις του νευρωνικού

5. Πατώντας το button 5 ανοίγει ένα παράθυρο (Σχήμα 7.9) με τις όλες τις κινήσεις που μπορεί να παίξει ο παίκτης στο αμέσως επόμενο βήμα του παιχνιδιού μαζί με την αξία των επιμέρους κινήσεων. Υπάρχει επίσης και η πρόταση του νευρωνικού προς τον παίκτη που είναι συνήθως η κίνηση με τη μεγαλύτερη αξία αφού κάνουμε exploitation στο 90% των περιπτώσεων και exploration στο υπόλοιπο 10%.



Σχήμα 7.9 Όλες οι πιθανές επόμενες κινήσεις του παίκτη και η αξία τους μαζί με την πρόταση του νευρωνικού

6. Το button 6 αντιστοιχεί στη βοήθεια. Πατώντας το ανοίγει ένα νέο παράθυρο (Σχήμα 7.10) που περιέχει πληροφορίες για τους κανόνες του παιχνιδιού και επεξηγήσεις κάποιων βασικών όρων που χρησιμοποιούνται στο παιχνίδι.



Σχήμα 7.10 Το παράθυρο βοήθειας με ππληροφορίες για τα συστατικά του παιχνιδιού και τους κανόνες κίνησης των πιονιών.

8. Συμπεράσματα - Επεκτάσεις της διπλωματικής εργασίας

Η παρούσα διπλωματική εργασία είχε ως αντικείμενο τη μοντελοποίηση των παικτών σε ένα παιχνίδι στρατηγικής και τη βελτίωση της συμπεριφοράς τους. Στα πλαίσια της διπλωματικής μελετήθηκαν και παρουσιάστηκαν σημαντικοί τομείς της Τεχνητής Νοημοσύνης όπως η Ενισχυτική Μάθηση και τα Νευρωνικά δίκτυα και βέβαια η έννοια της μοντελοποίησης στα παιχνίδια στρατηγικής και οι πιθανοί τρόποι μοντελοποίησης ενός παίκτη. Υλοποιήθηκε μάλιστα μία τέτοια μοντελοποίηση αν και το επίπεδο των γνώσεων του νευρωνικού δε μας επέτρεψε να διαπιστώσουμε τα αποτελέσματα της στην πράξη. Στο σημείο αυτό, έχοντας κατά κάποιο τρόπο τελειώσει με τη διπλωματική αξίζει να τονίσουμε κάποια βασικά συμπεράσματα που προέκυψαν και να αναφερθούμε σε πιθανές βελτιώσεις και προεκτάσεις που θα αυξήσουν την αξιοπιστία, την ευχρηστία και την λειτουργικότητα του παιχνιδιού.

8.1. Βασικά συμπεράσματα

Η ενισχυτική μάθηση αποτελεί έναν πολύ ενδιαφέροντα τομέα της μηχανικής μάθησης λόγω της γενικότητάς της και του μεγάλου πλήθους προβλημάτων στα οποία μπορεί να εφαρμοστεί. Είναι πραγματικά εντυπωσιακό το γεγονός ότι κάποιο σύστημα μπορεί να μάθει να προσομοιώνει τη συμπεριφορά που θέλουμε αλληλεπιδρώντας με το περιβάλλον του και εξετάζοντας την αμοιβή που έλαβε για τις διάφορες κινήσεις του.

8.2. Βελτίωση νευρωνικού

Η μέχρι στιγμής απόδοση του νευρωτικού δεν είναι καλή γεγονός που περιορίζει τις δυνατότητες του παιχνιδιού. Θα πρέπει να εξεταστεί εκ νέου η δομή του νευρωνικού δικτύου και να ανακαλυφθεί η αιτία. Πιθανές αιτίες θα μπορούσαν να είναι το πλήθος των νευρώνων του επιπέδου εισόδου ή του κρυφού επιπέδου και οι διάφοροι παράμετροι του νευρωνικού όπως ο ρυθμός μάθησης. Όλες αυτές οι εκδοχές ωστόσο είναι υποθετικές και θα πρέπει να εξεταστούν προκειμένου να βρεθεί η πραγματική αιτία.

8.3. Εκπαίδευση δικτύου –Πειράματα

Μόλις πριν αναφερθήκαμε στο νευρωνικό και είπαμε πως το γεγονός ότι δε μαθαίνει πιθανόν να οφείλεται στη δομή του. Ωστόσο θα μπορούσε να φταίει και η μη επαρκής εκπαίδευση του δικτύου. 50000 πειράματα δεν είναι αρκετά για την εκπαίδευση ενός νευρωνικού δικτύου που χρησιμοποιεί τον αλγόριθμο της προς τα πίσω διάδοσης *(back propagation algorithm)* που είναι ιδιαίτερα αργός. Αξίζει να αναφέρουμε εδώ πως ο Tesauro, το TD-Gammon του οποίου αποτελεί το πιο τρανταχτό παράδειγμα της επιτυχημένης εφαρμογής Ενισχυτικής Μάθησης, χρειάστηκε να τρέξει 1.500.000 παιχνίδια για την έκδοση TD-Gammon 2.1. Όπως αναφέρουν μάλιστα και οι [Sutton & Barto 1998] στην αρχή της εκπαίδευσης η αξία των κινήσεων του παιχνιδιού ήταν πολύ μικρή και τα παιχνίδια διαρκούσαν πολύ (εκατοντάδες – χιλιάδες κινήσεις) πριν κάποιος από τους δύο παίκτες κερδίσει πιθανόν τυχαία. Το ίδιο περίπου συμβαίνει και στη δική μας περίπτωση, γεγονός που μας επιτρέπει να εικάζουμε πως πιθανόν να ευθύνεται η μικρή εκπαίδευση για το ότι το παιχνίδι δεν μαθαίνει.

Κατά συνέπεια θα πρέπει να μελετηθεί πλήρως η δυναμική του παιχνιδιού για διάφορες διαστάσεις της σκακιέρας και για διάφορες τιμές των παραμέτρων μάθησης και των παραμέτρων του νευρωνικού δικτύου.

8.4. Γραφικό περιβάλλον

Το τρέχον γραφικό περιβάλλον του παιχνιδιού σχεδιάστηκε και υλοποιήθηκε κυρίως για διευκόλυνση στα πλαίσια της ανάπτυξης της διπλωματικής και το λόγο αυτό είναι ιδιαίτερα απλό. Στις μέρες μας όμως η

εμφάνιση και η ευχρηστία ενός περιβάλλοντος παίζουν σημαντικό ρόλο στην αποδοχή του περιβάλλοντος από τους τελικούς χρήστες και επηρεάζουν σε μεγάλο βαθμό την επιτυχία του. Για το λόγο αυτό θα πρέπει να σχεδιαστεί και να υλοποιηθεί ένα πλήρες γραφικό περιβάλλον για το χρήστη στηριζόμενο στο ήδη υπάρχον περιβάλλον.

Το νέο αυτό περιβάλλον θα πρέπει να κάνει πράξη τις αρχές του user interface, δηλαδή να είναι ελκυστικό όσον αφορά στην εμφάνισή του, εύκολο στην πλοήγηση και λειτουργικό ως προς τη χρήση του.

8.5. Μεταφορά στο διαδίκτυο

Στην παρούσα φάση οι πραγματικοί παίκτες (άνθρωποι δηλαδή) του παιχνιδιού είναι λίγοι γεγονός που δε μας επιτρέπει να βγάλουμε γενικά συμπεράσματα. Μια πιθανή μεταφορά του παιχνιδιού στο διαδίκτυο θα μπορούσε να λύσει το πρόβλημα και να μας προσφέρει επαρκές υλικό για παραπέρα μελέτη της συμπεριφοράς των παικτών. Αυτό προϋποθέτει ένα πλήρες γραφικό περιβάλλον στο οποίο αναφερθήκαμε στην προηγούμενη παράγραφο και έναν καλύτερο τρόπο διαχείρισης των παικτών. Στην παρούσα φάση τα στοιχεία των παικτών, τα επιμέρους παιχνίδια και τα στατιστικά των παιχνιδιών αποθηκεύονται σε αρχεία, γεγονός που θα πρέπει να αλλάξει για λόγους καλύτερης διαχείρισης. Μια πιθανή λύση είναι η δημιουργία μιας βάσης που θα περιλαμβάνει τουλάχιστον τρεις βασικούς πίνακες δεδομένων: έναν πίνακα με τους παίκτες, έναν πίνακα με τα παιχνίδια και έναν με τα μοντέλα των παικτών. Η βάση θα μπορούσε να είναι σε ORACLE και η διασύνδεση μαζί της να γίνεται μέσω του JDBC driver της JAVA δεδομένου ότι όλο το παιχνίδι είναι σε JAVA. Η μεταφορά στο διαδίκτυο προϋποθέτει την επιτυχή εκπαίδευση του νευρωνικού δικτύου προκειμένου να υπάρχουν κάποια bench βάρη που θα χρησιμοποιηθούν στα παιχνίδια με πραγματικούς παίκτες αντιπάλους.

9. Βιβλιογραφία

1. [Sutton & Barto 1998] R.S. Sutton & A.G. Barto. "*Reinforcement Learning - An Introduction",* MIT Press, Cambridge, Massachusetts, 1998.
2. [Teasuro 1995] G. Teasuro. "*Temporal Difference Learning and TD-Gammon",* Communications of the ACM, March 1995/Vol. 38, No 3.
3. [Sutton 1988] R.S. Sutton. "*Learning to Predict by the Methods of Temporal Differences",* Machine Learning 3: 9-44, 1988.
4. [Mitchell] T. M. Mitchell. "*Machine Learning",* The McGraw-Hill Companies, Inc.
5. [Singh & Sutton 1994] Singh, S.P., Sutton, R.S. "*Reinforcement learning with replacing eligibility traces",* Machine Learning 22: 123-158, 1996.
6. [Kalles & Kanellopoulos 2001] D. Kalles, P. Kanellopoulos. "*On verifying game design and playing strategies using reinforcement learning"*, ACM 2001.
7. [Tesauro 1992] G. Tesauro. "*Practical issues in temporal difference learning",* Machine Learning 8, 257-277, 1992
8. [Samuel 1959] A. Samuel. "*Some Studies in Machine Learning Using the Game of Checkers",* IBM Journal of Research and Development 3, 210-229, 1959
9. [Shanon 1950] C.E. Shanon. "*Programming a computer for playing chess"* Philosophical Magazine 41, 256-275, 1950.
10. [Knuth &Moore 1975] D. E. Knuth, R.E. Moore. "*An analysis of alpha beta pruning",* Artificial Intelligence 6 (4), 293-326, 1975.
11. [Leouski 1995] A. Leouski. "*Learning of Position Evaluation in the Game of Othelo",* Master' project: CMPSCI 701, University of Massachusetts, Amherst, 1995.
12. [Thrun 1995] S. Thrun. *Learning to Play the Game of Chess.* Advances in Neural Information Processing Systems 7, 1995.
13. [Λυκοθανάσης 1999] Σ. Λυκοθανάσης. *Υπολογιστική Νοημοσύνη Ι (Νευρωνικά Δίκτυα: Θεμελιώσεις και εφαρμογές)*. Πανεπιστήμιο Πατρών, Πολυτεχνική Σχολή, Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής, 1999.
14. S. Singh, P. Norvig, D. Cohn. "*How to make SoftWare Agents Do the Right Thing: An Introduction to Reinforcement Learning*", Adaptive Systems Group, Harlequin Inc., 1996
15. R.Givan, S.Leach, T.Deab. *"Bounded-parameter Markov-decision processes"*, Artificial Intelligence 122 (2000) 71-100.
16. M. Walker. *"An Application of Reinforcement Learning to Dialogue Strategy Selection in a Spoken Dialogue System for Email"*, Journal of Artificial Intelligence Research 12 (2000), 387-416.
17. S.Singh. "Learning to solve Markovian Decision Processes", Doctora thesis 1991, Department of Computer Science, University of Massachusetts.
18. [Gurney] K. Gurney. *Computers and Symbols versus Nets and Neurons*, Dept Human Sciences, Brunel University, Uxbridge, Middx.

19. [Gurney] K. Gurney. *The delta rule*, Dept Human Sciences, Brunel University, Uxbridge, Middx.

20. [Demuth & Beale] H. Demuth, M. Beale. *Neural Network Toolbox, For use with Matlab*.

21. [Balavanovic 97] M. Balabanovic and Y. Shoham, "Fab: Content-Based, Collaborative Recommendation", Communications of the ACM, Vol. 40, No. 3, March 1997, pp. 66-72.

22. [Krulwich 97] B. Krulwich and C. Burkey, "The InfoFinder Agent: Learning User Interests through Heuristic Phrase Extraction", IEEE Expert, Vol. 12, No. 5, Sep./Oct. 1997, pp. 22-27.

23. [Terveen 97] Terveen, W. Hill, B. Amento, D. McDonald, J. Creter, "PHOAKS: A System for Sharing Recommendations", Communications of the ACM, Vol. 40, No. 3, March 1997, pp. 59-62.

24. [Beck 1997] Joseph Beck *"Modeling the students with Reinforcement Learning" (http://www.dfki.uni-sb.de/~bauer/um-ws/Final-Versions/Beck/um_workshop.html)* in the Sixth International Conference on User Modeling, 1997

25. [Baffes 1994] Paul Baffes *"Automatic Student Modeling and Bug Library Construction using Theory Refinement"*, Ph.D. Thesis, Department of Computer Sciences, University of Texas at Austin, 1994.

26. [Baffes 1996] Paul Baffes and Raymond J. Mooney, "*A Novel Application of Theory Refinement to Student Modeling"*, *P*roceedings of the Thirteenth National Conference on Aritificial Intellige nce*, 1996(AAAI-96)*

27. [Baffes & Mooney 1996] Paul Baffes and Raymond J. Mooney, *"Refinement-Based Student Modeling and Automated Bug Library Construction",* Journal of Artificial Intelligence in Education*,1996*

28. [Witten & Frank 2000] Ian Witten and Eibe Frank, *"WEKA – Machine Learning Algorithms in Java",* Department of Computer Science, University of Waikato, New Zealand, Morgan Kaufmann Publishers*, 2000.*

29. Weka Data Mining System, Weka Experiment Environment, *2001*

30. J. Cleary, G. Holmes, S. Cunningham, I. Witten, *"MetaData for Database Mining" (*http://www.computer.org/conferences/meta96/holmes/DataBaseMining.html*),* Department of Computer Science, University of Waikato, New Zealand, IEEE 1996.

31. F. Meyer, *"Java Black Jack and Reinforcement Learning" (*http://lslwww.epfl.ch/ãperez/BlackJack/classes/RLJavaBJ.html*)*, Logic Systems Laboratory, EPFL, 1998.

10. Παράρτημα

10.1. Γλωσσάριο όρων

Το περιβάλλον μάθησης *(environment)*: Είναι το εξωτερικό σύστημα, τις καταστάσεις του οποίου παρατηρεί ο πράκτορας και αναλαμβάνει ανάλογες κινήσεις. Περιγράφεται από ένα σύνολο καταστάσεων *s*.

Ο μαθητής-πράκτορας *(agent):* Είναι ένα σύστημα που μαθαίνει μέσω των κινήσεων που αναλαμβάνει και οι οποίες αλλάζουν τις καταστάσεις του περιβάλλοντος. Παραδείγματα πρακτόρων είναι τα ρομπότ, οι πράκτορες λογισμικού, οι βιομηχανικοί ελεγκτές κ.α.

Οι καταστάσεις του περιβάλλοντος *(states of the environment):* Μπορούν να θεωρηθούν σαν μια περίληψη του παρελθόντος του συστήματος που καθορίζουν τη μελλοντική του εξέλιξη.

Η στρατηγική *(policy):* Αντικατοπτρίζει τον τρόπο με τον οποίο αποφασίζει ο πράκτορας, δηλαδή πως αναλαμβάνει κάποια κίνηση δοθείσας της τρέχουσας κατάστασης του περιβάλλοντος.

Η αμοιβή / τιμωρία *(reward):* Πρόκειται για μια μορφή αξιολόγησης των καταστάσεων του συστήματος που εκφράζει κατά πόσο η συγκεκριμένη κατάσταση είναι επιθυμητή ή όχι για τον πράκτορα.

Μαρκοβιανές διαδικασίες *(Markov decision processes):* Είναι εκείνες οι διαδικασίες όπου η τρέχουσα κατάσταση αντανακλά και όλες τις προηγούμενές της. Συνεπώς η επόμενη κίνηση εξαρτάται μόνο από την τρέχουσα κατάσταση του περιβάλλοντος.

Παράγοντας ρυθμού μείωσης *(discount factor)* γ: Είναι μία σταθερά που παίρνει τιμές στο διάστημα [0, 1] και καθορίζει την αξία των μελλοντικών αμοιβών *(rewards)*. Αν γ=0, ο πράκτορας ενδιαφέρεται για την άμεση μεγιστοποίηση της αμοιβής του. Αν γ=1, ο πράκτορας ενδιαφέρεται για την μακροπρόθεσμη μεγιστοποίηση της αμοιβής του.

Προσέγγιση συνάρτησης *(function approximation)*: Είναι μία συνάρτηση που προκύπτει από εκπαιδευτικά παραδείγματα τα οποία και περιγράφει. Το πρόβλημα της προσέγγισης μιας συνάρτήσης μπορεί να αντιμετωπιστεί με διάφορες μεθόδους Μηχανικής Μάθησης όπως δέντρα απόφασης, νευρωνικά δίκτυα κ.α.

Δυναμικός Προγραμματισμός *(Dynamic Programming)*: Χρησιμοποιείται για την επίλυση του προβλήματος της Ενισχυτικής Μάθησης. Πρόκειται για μία συλλογή αλγορίθμων που μπορούν να χρησιμοποιηθούν για τον υπολογισμό της βέλτιστης στρατηγικής *(optimal policy)* στην περίπτωση που το είναι γνωστό το πλήρες μοντέλο του περιβάλλοντος. Αν και η θεωρητική τους θεμελίωση είναι πολύ δυνατή δεν χρησιμοποιούνται συνήθως στην πράξη λόγω του περιορισμού που επιβάλλουν ως προς το μοντέλο του περιβάλλοντος και της μεγάλης τους πολυπλοκότητας.

Μέθοδοι Monte Carlo: Όπως και οι μέθοδοι Δυναμικού Προγραμματισμού, χρησιμοποιούνται για τον υπολογισμό της βέλτιστης στρατηγικής αλλά δεν απαιτούν πλήρη γνώση του μοντέλου του περιβάλλοντος. Προϋποθέτουν εμπειρία σε μορφή ακολουθιών τη μορφής: κατάσταση - κίνηση - αμοιβή *(reward)* που προέρχεται από άμεση ή έμμεση επαφή με το περιβάλλον. Παρόλο που είναι απλές και δεν απαιτούν γνώση του μοντέλου του περιβάλλοντος οι μέθοδοι Monte Carlo δεν είναι κατάλληλοι για αυξητικούς *(incremental)* υπολογισμούς.

Επιβλεπόμενη Μάθηση *(Supervised Learning)* : Στην επιβλεπόμενη μάθηση υπάρχει ένας εξωτερικός δάσκαλος ο οποίος θεωρείται γνώστης του περιβάλλοντος που αναπαρίσταται από ένα σύνολο παραδειγμάτων εισόδου-εξόδου. Το περιβάλλον ωστόσο είναι άγνωστο στο νευρωνικό δίκτυο. Ο

δάσκαλος εφοδιάζει το νευρωνικό με μια επιθυμητή απόκριση που παριστάνει την καλύτερη δυνατή δράση για το δίκτυο. Οι παράμετροι του δικτύου προσαρμόζονται κάτω από την κοινή επίδραση του διανύσματος μάθησης και του σήματος λάθους που ορίζεται ως η διαφορά στην πραγματική και την επιθυμητή απόκριση του δικτύου. Αυτή η προσαρμογή συνεχίζεται βήμα-βήμα με στόχο το νευρωνικό να συναγωνίζεται το διδάσκαλο, δηλαδή η γνώση του περιβάλλοντος που κατέχει ο δάσκαλος να μεταφερθεί στο νευρωνικό όσο πληρέστερα γίνεται. Όταν φτάσουμε σ' αυτό το σημείο μπορούμε να απαλλάξουμε το δάσκαλο και να αφήσουμε το δίκτυο να σχετίζεται μόνο του με το περιβάλλον.

10.2. Ευρετήριο σχημάτων & πινάκων

Ευρετήριο σχημάτων

Ευρετήριο πινάκων