Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Χημικών Μηχανικών

Τομέας ΙΙ: Ανάλυσης, Σχεδιασμού και Ανάπτυξης Διεργασιών και Συστημάτων

Διπλωματική Εργασία:

Ανάπτυξη Μεθοδολογίας Αυτόματης Ρύθμισης Συστημάτων με Εκπαίδευση Νευρωνικών Δικτύων Βάθους

Στεφανής Μιχαήλ
Επιβλέπων: Χαράλαμπος Σαρίμβεης

Σεπτέμβριος, 2019

Περιεχόμενα

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι η *ανάπτυξη μεθοδολογίας αυτόματης ρύθμισης συστημάτων με την χρήση τεχνικών τεχνητής νοημοσύνης*. Η εφαρμογή της προτεινόμενης μεθοδολογίας υλοποιείται σε γλώσσα προγραμματισμού Python με την βοήθεια της βιβλιοθήκης ανοιχτού κώδικα Tensorflow.

Στο πρώτο κομμάτι της εργασίας (κεφ. 1 έως και 6) περιγράφεται και αναλύεται το θεωρητικό υπόβαθρο που χρειάζεται κανείς, προκειμένου να είναι σε θέση να κατανοήσει τις επιμέρους συνιστώσες της μεθοδολογίας, και σχετίζεται με την θεωρία των νευρωνικών δικτύων και της ενισχυτικής μάθησης.

Το δεύτερο κομμάτι της εργασίας (κεφ. 7 έως και 9) αποτελείται από την αναλυτική περιγραφή του προτεινόμενου αλγορίθμου ρύθμισης καθώς και από εφαρμογές σε πραγματικές διεργασίες. Αρχικά γίνεται η εφαρμογή σε ένα γραμμικό σύστημα μίας εισόδου και μίας εξόδου, το οποίο χρησιμοποιούμε κυρίως σαν πρόβλημα αναφοράς για να ελέγξουμε κατά πόσο η μεθοδολογία μπορεί να ανταπεξέλθει σε ένα σχετικά απλό πρόβλημα ρύθμισης. Σε αυτήν την πρώτη εφαρμογή εξετάζουμε επίσης το είδος της επίδρασης που έχουν οι λεγόμενες συναρτήσεις κέρδους στην ρύθμιση του συστήματος. Στην συνέχεια εφαρμόζουμε την μεθοδολογία σε ένα μη-γραμμικό σύστημα δύο εισόδων και δύο εξόδων. Το σύστημα αυτό είναι ένας μη-ισοθερμοκρασιακός αντιδραστήρας CSTR. Στο πρόβλημα αυτό εξετάζουμε ποια είναι η επίδραση του θορύβου κατά Ornstein-Uhlenbeck στην ρύθμιση της διεργασίας. Παρουσιάζονται τα αποτελέσματα των προσομοιώσεων καθώς και σχολιασμοί και συμπεράσματα.

Abstract

The purpose of this thesis is the *development of a methodology for process control using techniques that come from the field of artificial intelligence*. The suggested methodology was applied using Python with the aid of the open-source code Tensorflow.

In the first part of the thesis (chapter 1 to 6) we describe and analyse the necessary theoretical background needed for the complete understanding of the various components of the methodology which is related with the theory of neural networks and reinforcement learning.

The second part of the thesis (chapter 7 to 9) is composed of a thorough description of the suggested control algorithm as well as applications in real processes. Firstly, we apply the algorithm to a linear one-input-one-output system, which we mainly use as a benchmark problem in order to check whether or nor the suggested methodology can cope with a fairly simple control problem. During this first application we also examine the effects of the so-called reward function to the process control. Thereinafter, we apply the methodology to a non-linear two-inputs-two-outputs system. This system is

a non-isothermal CSTR reactor. During this application we examine the effects of the Ornstein-Uhlenbeck noise to the process control. The results of the simulations as well as various comments are presented and discussed here.

Ευχαριστίες

Ευχαριστώ τον Παντελή Καρατζά και τον επιβλέπων καθηγητή Χαράλαμπο Σαρίμβεη για την βοήθεια και την καθοδήγηση σε όλη την διάρκεια της διπλωματικής εργασίας.

1 Εισαγωγή

Στην εργασία αυτή εξετάζεται η δυνατότητα σχεδιασμού ενός ρυθμιστή, του οποίου η λειτουργία να βασίζεται τόσο στην *Εκπαίδευση Νευρωνικών Δικτύων Βάθους* (Deep Learning) όσο και στην θεωρία της *Ενισχυτικής Μάθησης* (Reinforcement Learning). Η προτεινόμενη μεθοδολογία, ονόματι *Βαθμίδα Ντετερμινιστικής Πολιτικής Βάθους* (Deep Deterministic Policy Gradient, DDPG) [1], αναπτύχθηκε από την DeepMind το 2016 με σκοπό την ρύθμιση συστημάτων που χαρακτηρίζονται από συνεχές πεδίο τιμών των μεταβλητών εισόδου και εξόδου. Η εν λόγω προσέγγιση ρύθμισης βασισμένη στον αλγόριθμο DDPG έχει ένα μεγάλο πλεονέκτημα σε σχέση με τα συμβατικά συστήματα ρύθμισης, όπως ο PID και η LQR. Τα συστήματα αυτά προκειμένου να λειτουργήσουν έχουν ανάγκη ένα μοντέλο της διεργασίας. Είναι δηλαδή Model-Based συστήματα ρύθμισης. Το προτεινόμενο σύστημα ρύθμισης όμως δεν απαιτεί την ύπαρξη ενός μοντέλου της διεργασίας επειδή βασίζεται στην *εξερεύνηση* του χώρου μέσα στον οποίο ζει η διεργασία (ο οποίος φυσικά περιγράφεται από κάποιες εξισώσεις). Αυτή είναι η ριζική διαφορά που κάνει το εν λόγω σύστημα ρύθμισης τόσο ελκυστικό. Έχοντας μηδενική γνώση της διεργασίας και του μοντέλου της μπορούμε να προσεγγίσουμε μία επιθυμητή μόνιμη κατάσταση εξερευνώντας και *μαθαίνοντας* το περιβάλλον.

Η πορεία μέχρι να είμαστε σε θέση να παρουσιάσουμε τον βασικό αλγόριθμο ρύθμισης, DDPG, είναι η εξής. Θα αρχίσουμε παρουσιάζοντας κάποια γενικά στοιχεία που αφορούν την μηχανική μάθηση σαν επιστήμη, εφόσον τα νευρωνικά δίκτυα και η ενισχυτική μάθηση υπάγονται σε αυτήν. Στην συνέχεια θα εξετάσουμε κάποια πολύ βασικά στοιχεία του κλάδου της ρύθμισης διεργασιών έτσι ώστε να μπορέσουμε να τοποθετηθούμε καλύτερα στο τελικό μας πρόβλημα ρύθμισης. Έπειτα θα γίνει μία πολύ αναλυτική περιγραφή της εκπαίδευσης ενός νευρωνικού δικτύου βάθους (Deep Learning) καθώς και μία εισαγωγή στην θεωρία της Ενισχυτικής Μάθησης (Reinforcement Learning) αλλά και στις βασικές εξισώσεις που την συνιστούν, τις εξισώσεις Bellman. Έχοντας τις δύο αυτές βασικές θεωρίες θα τις συνδυάσουμε για να εξάγουμε την θεωρία της *Ενισχυτικής Μάθησης Βάθους* (Deep Reinforcement Learning), πάνω στην οποία βασίζεται και ο αλγόριθμος DDPG. Στην συνέχεια θα παρουσιάσουμε και να περιγράψουμε τον αλγόριθμο αλλά και να συνεχίσουμε με εφαρμογές, αποτελέσματα και σχολιασμούς. Στο τέλος της εργασίας παρατίθεται και ένα παράρτημα το οποίο είναι κυρίως μαθηματικού περιεχόμενου και περιέχει αποδείξεις διαφόρων θεωρημάτων και εξισώσεων που παρουσιάζονται ή/και χρησιμοποιούνται κατα μήκος όλης της εργασίας.

2 Μηχανική Μάθηση

2.1 Εισαγωγή

Η *Μάθηση*, όπως και η νοημοσύνη, καλύπτουν ένα πολύ ευρή φάσμα διαδικασιών που είναι δύσκολο να ορισθεί ακριβώς. Ορισμοί που συναντούμε συνήθως σε ένα λεξικό ή στο διαδίκτυο περιλαμβάνει φράσεις όπως "η απόκτηση γνώσεων και ικανοτήτων μέσα από μελέτη, οδηγίες ή εμπειρίες". Υπάρχουν αρκετοί παραλληλισμοί μεταξύ της βιολογικής και της μηχανικής μάθησης. Πράγματι, πολλές τεχνικές της μηχανικής μάθησης εξάγονται από τις προσπάθειες ψυχολόγων να βελτιώσουν τις θεωρίες τους πάνω στην βιολογική μάθηση μέσα από υπολογιστικά μοντέλα [2]. Ακόμα πιο εντυπωσιακό είναι το αντίστροφο. Πολλές έννοιες και τεχνικές που χρησιμοποιούνται από ερευνητές της μηχανικής μάθησης μπορούν να δώσουν φως σε μυστήρια που σχετίζονται με την βιολογική μάθηση. Στο κεφάλαιο αυτό θα ασχοληθούμε μόνο με την μηχανική μάθηση.

Η Μηχανική Μάθηση είναι ένας υποκλάδος της επιστήμης των υπολογιστών που εξελήχθηκε από την μελέτη της αναγνώρισης προτύπων και υπολογιστικής θεωρίας μάθησης στην τεχνητή νοημοσύνη. Το 1959, ο Arthur Samuel όρισε την μηχανική μάθηση ως "Το αντικείμενο μελέτης που δίνει στους υπολογιστές την ικανότητα να μάθουν χώρις να προγραμματίζονται με ρητό και καθορισμένο τρόπο" [3]. Το αντικείμενο της μηχανικής μάθησης είναι η κατασκευή και η μελέτη δυναμικών αλγορίθμων που μπορούν να *μάθουν* και να *κάνουν προβλέψεις* με βάση κάποια δεδομένα. Αυτού του είδους οι αλγόριθμοι λειτουργούν κατασκευάζοντας ένα *μοντέλο* από δεδομένα εισόδου με σκοπό να κάνουν προβλέψεις και να παίρνουν αποφάσεις καθοδηγούμενα από τα ίδια τα δεδομένα αντί να ακολουθούν αυστηρά κάποιο στατικό πρόγραμμα. Πρόκειται δηλαδή για την υλοποίηση ενός *δυναμικού* κώδικα ο οποίος θα μπορεί να προσαρμόζεται σε νέα δεδομένα, όπως ακριβώς ένας άνθρωπος μαθαίνει να προσαρμόζεται σε νέες καταστάσεις με την βοήθεια της εμπειρίας του και των γνώσεων που έχει αποκτήσει στο παρελθόν.

Η μηχανική μάθηση είναι στενά συνδεδεμένη με την υπολογιστική στατιστική, μία μαθηματική θεωρία που εξειδικεύεται στις προβλέψεις και είναι με την σειρά της συνδεδεμένη με την μαθηματική έννοια της *βελτιστοποίησης*. Το εύρος των εφαρμογών της μηχανικής μάθησης είναι εκπληκτικά μεγάλο όμως η μηχανική μάθηση αριστεύει στις εφαρμογές στις οποίες ο σχεδιασμός ενός (στατικού) προγράμματος-αλγορίθμου είναι αδύνατος. Παραδείγματα εφαρμογών είναι η *οπτική αναγνώριση χαρακτήρων* (γραμμάτων και αριθμών), οι *μηχανές αναζήτησης*, και το *φιλτράρισμα ανεπιθύμητων μηνυμάτων* (spam filtering) [3].

2.2 Τύποι προβλημάτων μηχανικής μάθησης

Τα προβλήματα στην μηχανική μάθηση συνήθως κατηγοριοποιούνται σε τρεις μεγάλες κατηγορίες με βάση την φύση των δεδομένων που είναι διαθέσιμα στο σύστημα εκμάθησης. Οι τρεις αυτές κατηγορίες είναι οι εξής [3]:

- **Εποπτευόμενη Μάθηση** (Supervised Learning): Στον υπολογιστή παρουσιάζονται τα δεδομένα εισόδου αλλά και οι επιθυμητές έξοδοι, οι οποίες παρατίθενται από τον προγραμματιστή, και ο στόχος είναι να βρεθεί (να μαθευτεί) ένας κανόνας, μία δηλαδή *συνάρτηση*, η οποία θα συσχετίζει την είσοδο με την έξοδο. Πρόκειται δηλαδή για προβλήματα στα οποία γνωρίζουμε από που αρχίζουμε αλλά και που θέλουμε να καταλήξουμε. Χαρακτηριστικό παράδειγμα είναι η αναγνώριση χαρακτήρων [9], όπου δίνουμε στον υπολογιστή, για παράδειγμα, αφενός τυχαίους φυσικούς αριθμούς (από το 0 μέχρι το 9) που έχουν γραφτεί από ανθρώπινο χέρι και αφετέρου τους ίδιους αριθμούς που όμως έχουν γραφτεί από υπολογιστή (και είναι κατά μία έννοια "τέλειοι"). Στόχος της μηχανικής μάθησης είναι να καταφέρει εν τέλει ο υπολογιστής να δεχθεί έναν χαρακτήρα που είναι γραμμένος από άνθρωπο και να μπορέσει να μας δώσει τον αντίστοιχο σωστό χαρακτήρα στην οθόνη του.

- **Μη-Εποπτευόμενη Μάθηση** (Unsupervised Learning): Στην περίπτωση αυτή τα δεδομένα εξόδου δεν δίνονται και αφήνουμε τον υπολογιστή να βρει ο ίδιος μία εσωτερική δομή-μοτίβο στα δεδομένα εισόδου. Τα προβλήματα αυτά είναι κατά κανόνα πιο απαιτητικά, απαιτούν μεγαλύτερο όγκο δεδομένων και συνήθως πρόκειται περί προβλημάτων ομαδοποίησης.

- **Ενισχυτική Μάθηση** (Reinforcement Learning): Στην ενισχυτική μάθηση ένα πρόγραμμα του υπολογιστή αλληλεπιδρά με ένα δυναμικό περιβάλλον στο οποίο θα πρέπει να πετύχει έναν συγκεκριμένο στόχο (όπως, για παράδειγμα, να οδηγήσει ένα όχημα) χωρίς κάποιον εξωτερικό πληροφοριοδότη για να του πει κατά πόσο έχει φτάσει κοντά στον στόχο του. Ένα παράδειγμα εφαρμογής της ενισχυτικής μάθησης είναι η ικανότητα ενός υπολογιστή να μάθει να παίζει ένα παιχνίδι (για παράδειγμα σκάκι) καθώς παίζει με έναν αντίπαλο. Το εκπληκτικό χαρακτηριστικό της ενισχυτικής μάθησης είναι ότι ο υπολογιστής δεν απαιτείται να έχει καμία πληροφορία για το (στην συγκεκριμένη περίπτωση) παιχνίδι, δηλαδή για τους κανόνες του παιχνιδιού. Μαθαίνει ό,τι πληροφορία είναι απαραίτητη για να κερδίσει μέσα από την εμπειρία που κερδίζει από τα πολλά παιχνίδια [4].

Μία άλλη κατηγοριοποίηση προβλημάτων μηχανικής μάθησης προκύπτει ανάλογα με την *επιθυμητή έξοδο* του συστήματος μάθησης [3]:

- **Ταξινόμηση (Classification)**: Στα προβλήματα ταξινόμησης τα δεδομένα χωρίζονται σε δύο ή περισσότερες κλάσσεις και το σύστημα εκμάθησης πρέπει να παράξει ένα μοντέλο το οποίο να εισάγει μία άγνωστη είσοδο σε μία από αυτές τις κλάσσεις. Συνήθως πρόκειται για εποπτευόμενα προβλήματα. Το φιλτράρισμα ανεπιθύμητων μηνυμάτων ανήκει στα προβλήματα ταξινόμησης. Τα εισερχόμενα μηνύματα είναι η είσοδος και οι κλάσσεις είναι "Ανεπιθύμητο" και "Επιθυμητό". Ένα άλλο παράδειγμα είναι η ταξινόμηση ανθρώπων με βάση το φύλο. Σε μία κάμερα μπορεί να εισαχθεί ένα πρόγραμμα το οποίο θα αποφασίζει αν ένας άνθρωπος είναι "άντρας" ή "γυναίκα"

(οι δύο κλάσσεις) με βάση χαρακτηριστικά όπως τα μαλλιά, το μέγεθος των χεριών κ.α.

- **Παλινδρόμηση (Regression)**: Τα προβλήματα παλινδρόμησης είναι και αυτά εποπτευόμενα προβλήματα όμως, σε αντίθεση με τα προβλήματα ταξινόμησης, οι έξοδοι του συστήματος είναι συνεχής ποσότητες αντι για διακριτές. Ένα πολύ γνωστό παράδειγμα παλινδρόμησης είναι η λεγόμενη *γραμμική παλινδρόμηση* κατά την οποία, στις δύο διαστάσεις, προσπαθούμε να προσδιορίσουμε μία ευθεία γραμμή η οποία να ταιριάζει όσο το δυνατόν καλύτερα στα δεδομένα μάς.

- **Ομαδοποίηση (Clustering)**: Στα προβλήματα ομαδοποίησης ένα σύνολο δεδομένων εισόδου χωρίζεται σε ομάδες. Σε αντίθεση με την περίπτωση της ταξινόμησης, οι ομάδες-κλάσσεις δεν είναι γνωστές εκ των προτέρων, κάτι που καθιστά τα προβλήματα αυτά μη-εποπτευόμενα.

- **Εκτίμηση Πυκνότητας (Density Estimation)**: Το πρόβλημα σε αυτήν την περίπτωση είναι η εύρεση της κατανομής των δεδομένων εισόδου που ορίζονται σε κάποιον χώρο.

- **Μείωση Διαστατικότητας (Dimensionality Reduction)**: Στα προβλήματα αυτά τα δεδομένα εισόδου απλοποιούνται απεικονίζοντάς τα σε κάποιον χώρο μικρότερης διάστασης. Παράδειγμα τέτοιου προβλήματος είναι το λεγόμενο *Topic Modeling*. Σε ένα πρόγραμμα υπολογιστή δίνουμε μία λίστα από έγγραφα γραμμένα σε κάποια γλώσσα και το ζητούμενο του υπολογιστή είναι να βρει ποια έγγραφα πραγματεύονται παρόμοια θέματα.

3 Ρύθμιση Διεργασιών

3.1 Εισαγωγή

Εφόσον εξετάζεται η δυνατότητα εύρεσης παραμέτρων ενός ρυθμιστή βασισμένου σε μεθοδολογίες τεχνιτής νοημοσύνης, θα πρέπει να εξετάσουμε τα βασικά δομικά στοιχεία του κλάδου της ρύθμισης διεργασιών.

Γενικά με τον όρο "*ρύθμιση*" εννοούμε, εν γένει, την ανάλήψη και την οργάνωση δράσεως προς διατήρηση των επιθυμητών συνθηκών σε μία οποιαδήποτε διεργασία. Η διεργασία αυτή είναι δυνατόν να είναι μηχανική, χημική, ηλεκτρική ή/και συνδυασμός αυτών [5]. Αυτή η δράση είναι δυνατόν να πραγματοποιηθεί είτε μέσω ανθρώπινης παρέμβασης (χειρισμό) ή και με κάποιο μηχανισμό. Στις εφαρμογές, η ρύθμιση αποβλέπει στην επίτευξη των επιθυμητών συνθηκών λειτουργίας και, κυρίως, στην δυνατότητα επαναφοράς στις επιθυμητές αυτές συνθήκες, όταν υπάρξει κάποια διαταραχή, όπως και στην δυνατότητα επιτεύξεως μόνιμης καταστάσεως όταν ένα ή/και περισσότερα εξωτερικά αίτια τείνουν να δημιουργήσουν μεταβατικές συνθήκες. Προκειμένου να υπάρχει ρύθμιση μίας διεργασίας, χρησιμοποιούμε σε κάθε περίπτωση μετρητικά, ρυθμιστές και τελικά στοιχεία ρύθμισης. Οι βασικοί στόχοι της ρύθμισης αφορούν πλέον την συμπεριφορά του συνολικού συστήματος και είναι οι εξής [5]:

- **Ευστάθεια**: Στις περισσότερες εφαρμογές της ρύθμισης θέλουμε το σύστημά μας να "ισορροπίσει" σε μία μόνιμη κατάσταση η οποία να είναι *ευσταθής*. Στην ουσία αυτό σημαίνει ότι αν στο σύστημα που ισορροπεί ασκηθεί μία (απρόβλεπτη) επιβολή τότε αυτό θα πρέπει να μπορέσει να επανέλθει στην προηγούμενη κατάσταση ισορροπίας. Φυσικά ο ρυθμιστής σε καμία περίπτωση δεν θα πρέπει να χαλάει την ευστάθεια του συστήματος.

- **Απόσβεση των διαταραχών**: Όπως θα δούμε παρακάτω, οι διαταραχές είναι μεταβλητές του συστήματος τις οποίες ο άνθρωπος δεν μπορεί να ελέγξει και να ρυθμίσει. Το σύστημα της ρύθμισης θα πρέπει να μπορεί να ανταπεξέλθει στην ύπαρξη αυτών των διαταραχών και να ρυθμίσει το σύστημα στο οποίο εμφανίζονται. Αυτό σημαίνει ότι αν μία διαταραχή απομακρύνει το σύστημα από την κατάσταση ισορροπίας του τότε ο ρυθμιστής θα πρέπει να είναι σε θέση να το επαναφέρει.

- **Ικανότητα καθοδηγητικού ελέγχου**: Στις περισσότερες πραγματικές εφαρμογές ρύθμισης είναι συχνά επιθυμητό, είτε για λόγους ασφάλειας είτε για λόγους παραγωγικότητας, η μόνιμη κατάσταση του συστήματος να αλλάξει. Στις περιπτώσεις αυτές θα πρέπει ο ρυθμιστής να μπορεί να "καθοδηγήσει" με ασφάλεια και όσο το δυνατόν πιο γρήγορα το σύστημα στην νέα αυτή μόνιμη κατάσταση.

- **Ευρωστία**: Το σύστημα ρύθμισης οφείλει να μπορεί να αναπεξέλθει σε απρόβλεπτες καταστάσεις που σχετίζονται με εξωτερικά φαινόμενα, όπως οι καιρικές συνθήκες ή/και φαινόμενα όπως οι σεισμοί. Θα πρέπει, για παράδειγμα, να διαθέτει "εργαλεία"

τα οποία να μπορούν να ρυθμίσουν επιτυχώς το σύστημα ακόμα και μία πολύ ζεστή καλοκαιρινή μέρα, όπου η θερμοκρασία του περιβάλλοντος μπορεί να είναι αρκετά μεγαλύτερη από την θερμοκρασία στην οποία σχεδιάστηκε για να λειτουργεί το σύστημα.

- **Ομαλή λειτουργία**: Κατά την διάρκεια της ρύθμισης, ο ρυθμιστής παίρνει συνεχώς "αποφάσεις", οι οποίες έχουν κάποια άμεση επίδραση στο σύστημα, με σκοπό να το καθοδηγήσει σε κάποια επιθυμητή κατάσταση ισορροπίας. Αυτές οι αποφάσεις κρύβουν φυσικά από πίσω κάποιο μέγεθος με φυσική σημασία, όπως για παράδειγμα θερμοκρασία και ροή. Συνεπώς ένα από τα βασικά μελήματα του ρυθμιστικού συστήματος θα πρέπει να είναι η ελαχιστοποίηση του κινδύνου και της φθοράς του εξοπλισμού που μπορεί να επιφέρει ο τρόπος ρύθμισης.

Για να κατανοήσουμε καλύτερα την έννοια της ρύθμισης μίας διεργασίας και για να μπορέσουμε να εδραιώσουμε την μαθηματική της μοντελοποίηση θα χρειαστούμε κάποιους ορισμούς. Αρχίζουμε με την έννοια του μαθηματικού μοντέλου ενός συστήματος καθώς και με την ταξινόμηση των ειδών των μεταβλητών που το απαρτίζουν.

Σύστημα: Ως *σύστημα* ορίζεται ως ένα σύνολο από στοιχεία (φυσικά ή/και μηχανικά) που αλληλεπιδρούν μεταξύ τους επηρεάζοντας το ένα το άλλο και συναποτελούν μια ολοκληρωμένη ενότητα. Ένα σύστημα οριοθετείται από το περιβάλλον μέσα στο οποίο βρίσκεται.

Μαθηματικό Μοντέλο: Το *μαθηματικό μοντέλο* ενός συστήματος είναι στην ουσία το σύνολο των εξισώσεων (αλγεβρικών και διαφορικών) που το περιγράφουν. Οι εξισώσεις αυτές περιέχουν φυσικά μεταβλητές που μας βοηθούν να προσδιορίσουμε τις επιθυμητές ιδιότητες αλλά και την συμπεριφορά του συστήματος μία δεδομένη χρονική στιγμή. Αν σε ένα σύστημα περιλαμβάνονται διαφορικές εξισώσεις που περιγράφουν κάποιο μέγεθος, τότε οι ανεξάρτητες μεταβλητές ως προς τις οποίες λαμβάνονται οι παράγωγοι μπορεί να είναι ο χρόνος ή/και οι χωρικές μεταβλητές ανάλογα με το αν οι τιμές των μεταβλητών αλλάζουν με τον χρόνο ή/και αν μεταβάλλονται σε μία, δύο ή τρεις διαστάσεις. Αν υπάρχουν παράγωγοι ως προς τον χρόνο τότε το σύστημα χαρακτηρίζεται ως *δυναμικό*, αλλιώς ως *μη-δυναμικό* ή *στατικό*. Σε ένα στατικό σύστημα δεν υπάρχουν χρονικές παράγωγοι αλλά μπορεί να υπάρχουν μερικές παράγωγοι ως προς μία ή περισσότερες χωρικές διαστάσεις. Αν υπάρχουν παράγωγοι μόνο ως προς τον χρόνο τότε το σύστημα χαρακτηρίζεται ως *συγκεντρωτικό δυναμικό*. Αν υπάρχουν παράγωγοι και ως προς τον χρόνο και ως προς τουλάχιστον μία χωρική μεταβλητή τότε το σύστημα χαρακτηρίζεται ως *κατανεμημένο δυναμικό* και προφανώς το μοντέλο περιλαμβάνει μερικές διαφορικές εξισώσεις.

3.2 Μεταβλητές Συστήματος Ρύθμισης

Συνεχίζουμε με την ταξινόμηση των διαφόρων μεταβλητών που μπορεί να περιλαμβάνονται στο μαθηματικό μοντέλο μιας διεργασίας [5]:

- **Ρυθμιζόμενες Μεταβλητές (controlled variables)**: είναι οι μεταβλητές που θέλουμε να ρυθμίσουμε συνήθως λόγω κάποιας απαίτησης ή για λόγους ασφαλείας. Θα πρέπει να μπορούν να μετρηθούν άμεσα ή έμμεσα μέσω μέτρησης άλλων μεγεθών που σχετίζονται με αυτές. Σε ορισμένες περιπτώσεις κάποιες ρυθμιζόμενες μεταβλητές μπορεί να είναι και μεταβλητές εξόδου.

- **Μεταβλητές Εισόδου (input variables)**: είναι οι (ανεξάρτητες) μεταβλητές που περιγράφουν με ποιόν τρόπο το σύστημά μας αλληλεπιδρά με το περιβάλλον και πώς το ένα επηρεάζει το άλλο. Αν ο αριθμός των μεταβλητών εισόδου είναι N_I τότε το σύνολο των μεταβλητών εισόδου μπορεί να περιγραφεί από το διάνυσμα στήλη:

$$\mathbf{u}(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_{N_I}(t) \end{bmatrix}$$

Στην συνέχεια, οι μεταβλητές εισόδου διακρίνονται σε *διαταραχές*, *μεταβλητές εκ χειρισμού* και *σήματα αναφοράς*:

Διαταραχές (disturbances): είναι οι μεταβλητές εισόδου που ο άνθρωπος δεν μπορεί να επηρεάσει και να ρυθμίσει κατά την δυναμική τους πορεία στον χρόνο. Το μόνο που μπορεί να κάνει κάποιος είναι, όσο αυτές εκδηλώνονται, να τις μετρήσει. Οι μεταβλητές αυτές συνήθως προκαλούν αποκλίσεις της διεργασίας από την επιθυμητή κατάσταση λειτουργίας. Σε ένα σύστημα στο οποίο δεν λαμβάνει χώρα διαδικασία ρύθμισης, όλες οι μεταβλητές εισόδου είναι διαταραχές. Το σύνολο των N_D διαταραχών περιγράφεται με την βοήθεια του διανύσματος:

$$\mathbf{u}_D(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_{N_D}(t) \end{bmatrix}$$

Πίσω από κάθε διαταραχή u_i υπάρχει μία συνάρτηση ως προς τον χρόνο $u_i = f_i(t)$ που συνήθως είναι άγνωστη εκ των προτέρων. Με την μέτρηση ή την εκτίμηση μίας διαταραχής ουσιαστικά μαθαίνουμε την συνάρτηση $f_i(t)$.

Μεταβλητές εκ χειρισμού (manipulated inputs): είναι οι μεταβλητές εισόδου οι οποίες αξιοποιούνται από τον σχεδιαστή του συστήματος ρύθμισης, ώστε να επιτευχθούν οι στόχοι της ρύθμισης. Το σύνολο των N_M μεταβλητών χειρισμού περιγράφεται με την βοήθεια του διανύσματος:

$$\mathbf{u}_M(t) = \begin{bmatrix} u_{N_D+1}(t) \\ u_{N_D+2}(t) \\ \vdots \\ u_{N_D+N_M}(t) \end{bmatrix}$$

Ένα παράδειγμα μεταβλητής εκ χειρισμού θα μπορούσε να είναι η θερμοκρασία ή η ροή ενός ψυκτικού μέσου που χρησιμοποιείται για να ψύξει έναν αντιδραστήρα μέσα στον οποίον διαδραματίζεται μία εξώθερμη αντίδραση. Στόχος του ρυθμιστικού συστήματος είναι να βρεί τον βέλτιστο δυνατό τρόπο αξιοποίησης αυτών των μεταβλητών με σκοπό την επιτυχή ρύθμιση του συστήματος.

Σήματα αναφοράς ή επιθυμητές τιμές (set points): είναι οι μεταβλητές εισόδου οι οποίες περιγράφουν την τελική επιθυμητή κατάσταση του συστήματος. Μπορεί να είναι σταθεροί αριθμοί ή ακόμη και συναρτήσεις του χρόνου. Τα σήματα αναφοράς έχουν νόημα μόνο εφόσον έχει εγκατασταθεί ένα σύστημα ρύθμισης. Το σύνολο N_{SP} σημάτων αναφοράς περιγράφεται από το διάνυσμα:

$$\mathbf{u}_{SP}(t) = \begin{bmatrix} u_{N_D+N_M+1}(t) \\ u_{N_D+N_M+2}(t) \\ \vdots \\ u_{N_D+N_M+N_{SP}}(t) \end{bmatrix}$$

Από τους παραπάνω ορισμούς προφανώς ισχύει ότι $N_I = N_D + N_M + N_{SP}$. Συνεχίζουμε με την περιγραφή των υπόλοιπων ειδών μεταβλητών.

- **Μεταβλητές Κατάστασης (state variables)**: είναι οι εξαρτημένες μεταβλητές του μοντέλου μας και, όπως υποδηλώνει και το όνομά τους, μας περιγράφουν την κατάσταση του συστήματος μας μία δεδομένη χρονική στιγμή. Αποτελούν και αυτές φυσικά συναρτήσεις του χρόνου. Το σύνολο N_S των μεταβλητών κατάστασης περιγράφεται από το διάνυσμα:

  $$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_{N_S}(t) \end{bmatrix}$$

  Το διάνυσμα μεταβλητών κατάστασης προσδιορίζει *μονοσήμαντα* ένα σύστημα σε κάθε χρονική στιγμή όταν είναι γνωστά:

  (i) οι χρονικές συναρτήσεις όλων των μεταβλητών εισόδου, δηλαδή το διάνυσμα $\mathbf{u}(t)$
  (ii) Το διάνυσμα των αρχικών συνθηκών $\mathbf{x}(t_0) = \mathbf{x}_0$

  Ο λόγος που ισχύει αυτή η ικανότητα των μεταβλητών κατάστασης να μας δίνουν μονοσήμαντες πληροφορίες για ένα σύστημα είναι το γεγονός ότι το ίδιο το σύστημα περιγράφεται, όπως θα δούμε παρακάτω, από σύστημα *διαφορικών εξισώσεων* και όταν έχουμε στην διάθεσή μας τόσες αρχικές συνθήκες όσες και οι άγνωστες συναρτήσεις τότε η λύση του συστήματος θα είναι μοναδική.

- **Μεταβλητές Εξόδου (output variables)**: είναι οι ρυθμιζόμενες μεταβλητές που περιγράφουν με ποιον τρόπο το σύστημα επηρεάζει το περιβάλλον. Οι μεταβλητές εξόδου αποτελούν συναρτήσεις των μεταβλητών κατάστασης αλλά και των μεταβλητών εισόδου, κάτι που δεν θα πρέπει να μας κάνει εντύπωση αφού προφανώς η έξοδος του συστήματος εξαρτάται τόσο από την τρέχουσα κατάστασή του όσο και από την επίδραση που έχουν οι μεταβλητές εισόδου σε αυτό. Το σύνολο των N_O μεταβλητών εξόδου περιγράφεται από το διάνυσμα:

$$\mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_{N_O}(t) \end{bmatrix}$$

3.3 Χώρος Κατάστασης

Στην συνέχεια θα δούμε πώς όλα αυτά τα είδη μεταβλητών συνδέονται για να μοντελοποιήσουν το σύστημα μιας διεργασίας και κατ'επέκταση το σύστημα ρύθμισης.

Κατάσταση: *Κατάσταση* ενός δυναμικού συστήματος είναι το σύνολο των μεταβλητών (που όπως είδαμε και προηγουμένως καλούνται και μεταβλητές κατάστασης), η γνώση των τιμών των οποίων κάποια αρχική χρονική στιγμή $t = t_0$, καθώς και η γνώση όλων μεταβλητών εισόδου για $t \geq t_0$, προσδιορίζει πλήρως την συμπεριφορά του συστήματος για κάθε χρονική στιγμή $t \geq t_0$ [5].

Χώρος Κατάστασης: Έστω ένα σύστημα το οποίο περιγράφεται από το διάνυσμα μεταβλητών κατάστασης:

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) & x_2(t) & \cdots & x_N(t) \end{bmatrix}^\mathsf{T}$$

Τότε ο n-διάστατος διανυσματικός χώρος $\mathcal{X} = span[(x_1(t), x_2(t) \cdots x_N(t))] \subset \mathbb{R}^n$ καλείται *χώρος κατάστασης* του συστήματος. Κάθε πιθανή κατάσταση του συστήματος αντιστοιχεί μονοσήμαντα σε ένα σημείο του χώρου $\mathcal{X}$.

Μαθηματικό Μοντέλο Διεργασίας: Η γενική μορφή ενός μαθηματικού μοντέλου ενός συστήματος είναι η εξής:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \\ \mathbf{y}(t) &= \mathbf{g}(\mathbf{x}, \mathbf{u}, t) \end{aligned} \tag{3.3.1}$$

με δεδομένες τις αρχικές συνθήκες $\mathbf{x}(0) = \mathbf{x}_0$. Στην παραπάνω διατύπωση $\mathbf{x}$ είναι το διάνυσμα μεταβλητών κατάστασης, $\mathbf{u}$ το διάνυσμα των μεταβλητών εισόδου και $\mathbf{y}$ το διάνυσμα των μεταβλητών εξόδου. Η λύση των εξισώσεων (3.3.1) αναπαριστάται στον χώρο $\mathcal{X}$ με μία γραμμή που ξεκινά από το $\mathbf{x}_0$.

Χαρακτηριστική ειδική περίπτωση των εξισώσεων (3.3.1) αποτελούν τα λεγόμενα *γραμμικά συστήματα*. Στην περίπτωση αυτή οι εξισώσεις του συστήματος παίρνουν την μορφή:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}(t)\mathbf{x}(t) + \mathbf{D}(t)\mathbf{u}(t) \end{aligned} \tag{3.3.2}$$

όπου οι διαστάσεις των πινάκων $\mathbf{A}$,$\mathbf{B}$,$\mathbf{C}$,$\mathbf{D}$ είναι τέτοιες ώστε να συμβαδίζουν με τις διαστάσεις των $\mathbf{x}$,$\mathbf{y}$,$\mathbf{u}$. Αν επίσης οι πίνακες αυτοί εξαρτώνται από τον χρόνο τότε το σύστημα λέγεται *άμεσα εξαρτώμενο από τον χρόνο*, αλλιώς λέγεται *μη-εξαρτώμενο άμεσα από τον χρόνο*.

3.4 Γραμμικά Μοντέλα Εισόδου-Εξόδου και Συναρτήσεις Μεταφοράς

Σε ό,τι αφορά την συνέχεια αυτού του κεφαλαίου θα αναφερόμαστε σε γραμμικά και μη-εξαρτώμενα άμεσα από τον χρόνο συστήματα, δηλαδή συστήματα της μορφής:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \end{aligned} \tag{3.4.1}$$

όπου οι πίνακες $\mathbf{A}$,$\mathbf{B}$,$\mathbf{C}$,$\mathbf{D}$ δεν περιλαμβάνουν ως μεταβλητή τον χρόνο και οι αρχικές συνθήκες του συστήματος είναι $\mathbf{x}(0) = \mathbf{0}$.

Ας υποθέσουμε για αρχή ότι έχουμε ένα σύστημα μιας εισόδου και μίας εξόδου της μορφής (3.4.1). Στην περίπτωση αυτή τα διανύσματα $\mathbf{x}$ και $\mathbf{u}$ έχουν διάσταση 1×1, δηλαδή πρόκειται για πραγματικές συναρτήσεις ενώ οι πίνακες $\mathbf{A}$,$\mathbf{B}$,$\mathbf{C}$,$\mathbf{D}$ είναι απλώς πραγματικοί αριθμοί. Το πρώτο μας βήμα είναι να πάρουμε τους *μετασχηματισμούς Laplace* των εξισώσεων αυτών για να μεταβούμε από το πεδίο του χρόνου στο πεδίο των συχνοτήτων. Λαμβάνουμε:

$$\begin{aligned} s\mathbf{x}(t) &= \mathbf{A}\mathbf{x}(s) + \mathbf{B}\mathbf{u}(s) \\ \mathbf{y}(s) &= \mathbf{C}\mathbf{x}(s) + \mathbf{D}\mathbf{u}(s) \end{aligned} \tag{3.4.2}$$

Στην συνέχεια απαλείφουμε το $\mathbf{x}$(s) από τις δύο εξισώσεις και παίρνουμε την σχέση:

$$\mathbf{y}(s) = [\mathbf{C}(S\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}]\mathbf{u}(s)$$

Τότε ως *συνάρτηση μεταφοράς* του συστήματος ορίζεται η ρητή συνάρτηση:

$$\mathbf{G}(s) = \frac{\mathbf{y}(s)}{\mathbf{u}(s)} = [\mathbf{C}(S\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}] \tag{3.4.3}$$

Πρόκειται δηλαδή για το πηλίκο του κατά Laplace μετασχηματισμένου σήματος εξόδου προς το κατά Laplace μετασχηματισμένο σήμα εισόδου όταν οι αρχικές συνθήκες είναι ίσες με μηδέν [6]. Η παραπάνω εξίσωση δεν περιλαμβάνει πλέον μεταβλητές κατάστασης αλλά μόνο την μεταβλητή εισόδου και την μεταβλητή εξόδου. Είναι μία μορφή που υπάγεται στην γενική κατηγορία των μοντέλων εισόδου-εξόδου. Το συγκεκριμένο μοντέλο που συσχετίζει

την είσοδο $\mathbf{u}$ με την έξοδο $\mathbf{y}$ λέγεται , όπως προαναφέρθηκε, συνάρτηση μεταφοράς. Είναι δηλαδή η συνάρτηση που μας δείχνει πως η είσοδος του συστήματος επηρεάζει την έξοδο του συστήματος.

Στην περίπτωση των συστημάτων πολλαπλών εισόδων-πολλαπλών εξόδων η συλλογιστική πορεία εξαγωγής της συνάρτησης μεταφοράς είναι παρόμοια. Στην περίπτωση αυτή η ποσότητα:

$$\mathbf{G}(s) = [\mathbf{C}(S\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}]$$

δεν είναι πραγματική συνάρτηση αλλά ενας *πίνακας* που αποτελείται από πραγματικές συναρτήσεις. Πιο συγκεκριμένα, το (i, j) στοιχείο του πίνακα αυτού είναι η συνάρτηση μεταφοράς:

$$\mathbf{G}_{i,j}(s) = \frac{\mathbf{y}_i(s)}{\mathbf{u}_j(s)}$$

πρόκειται δηλαδή για την συνάρτηση που μας δείχνει πως η j είσοδος του συστήματος επηρεάζει την i έξοδο του συστήματος.

Η συνάρτηση μεταφοράς μας δίνει μία πληθώρα πληροφοριών για την διεργασία μας. Όχι μόνο μπορούμε να δούμε πώς μία μεταβλητή εισόδου επηρεάζει μία μεταβλητή εξόδου αλλά επίσης μπορούμε να αποφανθούμε σε πολλές περιπτώσεις κατά πόσο το σύστημά μας είναι ευσταθές ή όχι [5].

Τέλος σημειώνεται ότι για γραμμικά συστήματα με σταθερούς συντελεστές της μορφής (3.4.1) υπάρχει η αναλυτική λύση, η οποία είναι:

$$\mathbf{y}(t) = \mathbf{C}e^{\mathbf{A}t}\mathbf{x}_0 + \mathbf{C}\int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau + \mathbf{D}\mathbf{u}(t) \tag{3.4.4}$$

με την προϋπόθεση φυσικά ότι οι συναρτήσεις $\mathbf{u}$(t) είναι συνεχείς και φραγμένες (δηλαδή ομοιόμορφα συνεχές). Το εκθετικό ενός τετραγωνικού πίνακα $\mathbf{A}$ ορίζεται ως $e^{\mathbf{A}t} = \sum_{k=0}^{\infty} \frac{\mathbf{A}^k t^k}{k!}$.

4 Εκπαίδευση Νευρωνικών Δικτύων Βάθους (Deep Learning)

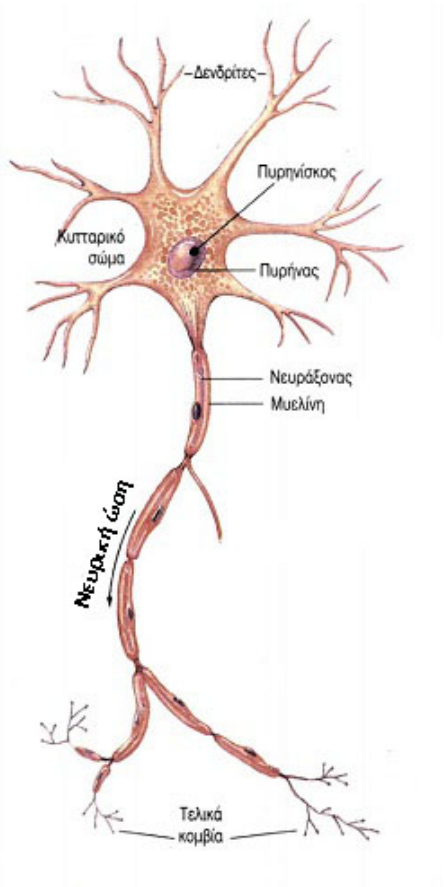
4.1 Εισαγωγή

Υπάρχουν κατηγορίες προβλημάτων που δεν μπορούν να διατυπωθούν σαν αλγόριθμοι. Τέτοια προβλήματα εξαρτώνται από πολλές αρκετά λεπτές παραμέτρους, οι οποίες συνήθως μεταβάλλονται με τον χρόνο. Χαρακτηριστικό παράδειγμα είναι η πρόβλεψη του καιρού, ένα χαοτικό σύστημα του οποίου η συμπεριφορά εξαρτάται από πολλές δυναμικές παραμέτρους, όπως η πίεση, το υψόμετρο, η τοποθεσία κ.α. Ένας άνθρωπος που έχει ζήσει αρκετά πάνω στην γη ξέρει ότι όταν δει πολλά σύννεφα στον ουρανό τότε κατά πάσα πιθανότητα θα βρέξει (χωρίς απαραίτητα να ξέρει τί προκάλεσε την δημιουργία των συννέφων ή για ποιο λόγο αρχίζει η βροχή) και γι'αυτό προετοιμάζεται κατάλληλα παίρνοντας ομπρέλα ή/και φορώντας περισσότερα ρούχα. Έχει *μάθει* δηλαδή να συσχετίζει τα σύννεφα με την βροχή, μέσα από τις εμπειρίες του, και έτσι μπορεί και προσαρμόζεται στις ενδεχόμενες αλλαγές του καιρού. Ένας υπολογιστής όμως απαιτεί έναν αλγόριθμο για να μπορέσει να κάνει το ίδιο, δηλαδή να συσχετίσει αλλαγές στο περιβάλλον με αλλαγές στον καιρό. Αυτή είναι και η ριζική διαφορά μεταξύ του ανθρώπινου εγκεφάλου και ενός υπολογιστή. Ότι ο ανθρώπινος εγκέφαλος *μαθαίνει* και *προσαρμόζεται*. Οι υπολογιστές μπορούν να εκτελέσουν πολύ σύνθετους υπολογισμούς σε πολύ λίγο χρόνο όμως δεν είναι καθόλου προσαρμοστικοί. Θα ήταν λοιπόν πολύ χρήσιμο αν μπορούσαμε να κάνουμε έναν υπολογιστή να *σκέφτεται* όπως ακριβώς σκέφτεται και ο ανθρώπινος εγκέφαλος. Στην επιθυμία αυτή βασίζεται η επιστήμη των νευρωνικών δικτύων. Θα θέλαμε δηλαδή να δημιουργήσουμε μία ψηφιακή δομή στον υπολογιστή η οποία να επεξεργάζεται δεδομένα με παρόμοιο τρόπο που τα επεξεργάζεται και ο ανθρώπινος εγκέφαλος. Δίνουμε λοιπόν έναν πρωταρχικό ορισμό ενος νευρωνικού δικτύου [7]:

Ορισμός: Ένα (τεχνητό) *νευρωνικό δίκτυο* είναι ένα διασυνδεδεμένο συγκρότημα από απλά στοιχεία επεξεργασίας, τους *κόμβους* ή *νευρώνες*, η λειτουργία των οποίων είναι βασισμένη στους νευρώνες των ανθρώπων. Η ικανότητα επεξεργασίας ενός νευρωνικού δικτύου είναι αποθηκευμένη στους ενδονευρωνικούς συνδέσμους, τα λεγόμενα *βάρη*, οι οποίοι προσδιορίζονται από μία διαδικασία προσαρμογής (ή *μάθησης*) σε ένα σύνολο από μοτίβα εκπαίδευσης.

Θα συνεχίσουμε κάνοντας μία μικρή αναφορά στα βασικά στοιχεία της νευροβιολογίας. Ο ανθρώπινος εγκέφαλος αποτελείται από περίπου 10^{11} (100 δισεκατομμύρια) νευρικά κύτταρα που λένονται *νευρώνες* [7]. Οι νευρώνες επικοινωνούν μεταξύ τους μέσω ηλεκτρικών μηνυμάτων τα οποία αποτελούν εφήμερους παλμούς της ηλεκτρικής τάσης του κυτταρικού τοιχώματος. Ανάμεσα στις ενδονευρωνικές συνδέσεις παρεμβάλλονται ηλεκτροχημικές διασυνδέσεις που λέγονται *συνάψεις*. Βρίσκονται στα "κλαδία" του νευρικού κυττάρου, τους λεγόμενους *δενδρίτες*. Κάθε νευρώνας δέχεται αρκετές χιλιάδες συνδέσεις από άλλους νευρώνες και συνεπώς δέχεται σε κάθε στιγμή μία πληθώρα από εισερχόμενα ηλεκτρικά σήματα, τα οποία εν τέλει φθάνουν στο κυρίως σώμα του κυττάρου. Εκεί, αθροίζονται με κάποιον τρόπο και, χονδρικά, αν το τελικό σήμα είναι μεγαλύτερο από μία συγκεκριμένη τιμή τότε ο νευρώνας θα παράγει έναν ηλεκτρικό παλμό. Ο παλμός αυτός τότε θα διαδοθεί σε

άλλους νευρώνες μέσω μίας ινώδους δομής γνωστής ώς *άξονας* ή *νευράξονας*. Η σχηματική αναπαράσταση ενός νευρικού κυττάρου φαίνεται παρακάτω



Εικόνα 1: Σχηματική Αναπαράσταση ενός Νευρικού Κυττάρου

Το αν ένα νευρικό κύτταρο θα εκπέμψει ένα ηλεκτρικό σήμα εξαρτάται από τα εισερχόμενα ηλεκτρικά σήματα. Κάποια από αυτά παράγουν μία παρεμποδιστική δράση και τείνουν να εμποδίσουν την εκπομπή του σήματος. Κάποια άλλα δρουν διεγερτικά και προάγουν την εκπομπή του ηλεκτρικού σήματος. Τελικά η ικανότητα του κάθε νευρικού κυττάρου να επεξεργάζεται δεδομένα βασίζεται στον τύπο (παρεμποδιστικό ή διεγετρικό) και στην δύναμη των συναπτικών συνδέσεων με άλλους νευρώνες.

Αυτού του είδους αρχιτεκτονική και στυλ επεξαργασίας δεδομένων επιθυμούμε να ενσωματώσουμε στα τεχνητά νευρωνικά δίκτυα. Το τεχνητό ισοδύναμο των βιολογικών νευρώνων είναι οι κόμβοι που εμφανίστηκαν στον αρχικό ορισμό του νευρωνικού δικτύου. Οι συνάψεις μοντελοποιούνται με έναν πραγματικό αριθμό, το *βάρος* (weight), έτσι ώστε κάθε είσοδος να πολλαπλασιάζεται με ένα βάρος πριν εισέλθει στο τεχνητό ισοδύναμο με το κυρίως σώμα

του νευρικού κυττάρου. Τα σταθμισμένα σήματα εισόδου αθροίζονται ακρίβως όπως γνωρίζουμε ότι αθροίζονται οι πραγματικοί αριθμοί και στην συνέχεια επεξεργάζονται από ειδικές συναρτήσεις, οι οποίες δίνουν το όριο με βάση το οποίο το κύτταρο αποφασίζει αν θα εκπέμψει το ηλεκτρικό του σήμα ή όχι.

4.2 Η μαθηματική δομή ενός Νευρωνικού Δικτύου

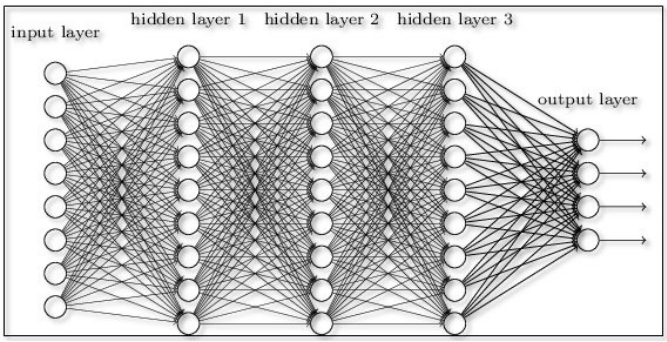
Η μαθηματική μοντελοποίηση ενός νευρωνικού δικτύου γίνεται με ένα *συνεκτικό κατευθυνόμενο γράφημα* (connected oriented graph). Αποτελείται δηλαδή από κόμβους (nodes), οι οποίοι καλούνται *νευρώνες* (neurons) και από *ακμές* (edges), οι οποίες συνδέουν τους νευρώνες μεταξύ τους. Η έννοια του συνεκτικού γραφήματος σημαίνει ότι κάθε νευρώνας θα πρέπει να συνδέεται με τουλάχιστον έναν άλλο νευρώνα μέσω μίας ακμής. Οι νευρώνες δεν τοποθετούνται τυχαία στο γράφημα αλλά έχουν μία συγκεκριμένη *δομή*. Πιο συγκεκριμένα, οι κόμβοι του γραφήματος κατανέμονται στα λεγόμενα *στρώματα* (layers) του δικτύου. Ένα τέτοιο στρώμα αποτελείται από κόμβους οι οποίοι *δεν επικοινωνούν μεταξύ τους* (δηλαδή δεν συνδέονται με ακμές) αλλά δέχονται πληροφορία από προηγούμενα στρώματα. Μπορούμε να θεωρήσουμε ένα στρώμα του νευρωνικού δικτύου με n νευρώνες ώς ένα διάνυσμα του $\mathbb{R}^n$.
Σε ένα νευρωνικό δίκτυο μπορούμε να διακρίνουμε τρία είδη στρωμάτων [9].

Στρώμα Εισόδου (Input Layer) : Το στρώμα αυτό αποτελεί την είσοδο του νευρωνικού δικτύου, με την έννοια ότι στο στρώμα αυτό τοποθετούνται τα δεδομένα προς εκπαίδευση. Τα δεδομένα αυτά, τα οποία μπορεί να προέρχονται από κάποια βάση δεδομένων (φωτογραφίες, κομμάτια ήχου κ.α.) ή ακόμη και να αποτελούν συναρτήσεις, προωθούνται στα επόμενα στρώματα του δικτύου με σκοπό να αρχίσει η εκπαίδευσή τους.

Κρυφά Στρώματα (Hidden Layers) : Τα στρώματα αυτά αποτελούν την καρδιά ενός νευρωνικού δικτύου καθώς σε αυτά γίνεται το μεγαλύτερο μέρος της εκπαίδευσης. Κάθε νευρώνας ενός κρυφού στρώματος συμβολίζει μία *συνάρτηση ενεργοποίησης* (activation function), οι οποίες θα αναλυθούν στην συνέχεια. Κάθε ακμή συμβολίζει το σήμα που μεταδίδεται από έναν νευρώνα στον επόμενο. Ο αριθμός των κρυφών στρωμάτων και ο αριθμός των νευρώνων σε κάθε κρυφό στρώμα αποτελεί θέμα κυρίως εμπειρικό και δεν υπάρχει ακόμη ιδιαίτερα κατατοπιστική μεθοδολογία για τον προσδιορισμό ενός βέλτιστου αριθμού κρυφών στρωμάτων και νευρώνων.

Στρώμα Εξόδου (Output Layer) : Στο στρώμα αυτό εισέρχονται τα εκπαιδευμένα δεδομένα που έχουν εξέλθει από το τελευταίο κρυφό στρώμα. Ο αριθμός των νευρώνων σε αυτό το στρώμα εξαρτάται από το εκάστοτε πρόβλημα.

Εφόσον το νευρωνικό δίκτυο σχηματίσει επιτυχώς το στρώμα εξόδου, σκοπός μας είναι να *συγκρίνουμε* τα δεδομένα του στρώματος εξόδου με τις προβλέψεις μας, δηλαδή με τις επιθυμητές τιμές. Αυτό επιτυγχάνεται με τη βοήθεια μιας ειδικής *συνάρτησης κόστους* (Loss Function). Η συνάρτηση αυτή μας βοηθά να ποσοτικοποιήσουμε το σφάλμα ή την απόκλιση των δεδομένων του στρώματος εξόδου σε σχέση με τις επιθυμητές τιμές. Στόχος



Εικόνα 2: Αναπαράσταση ενός νευρωνικού δικτύου με τρία κρυφά στρώματα

μας προφανώς είναι να ελαχιστοποιήσουμε το σφάλμα αυτό και σε αυτήν την ελαχιστοποίηση συνίσταται η *εκπαίδευση* ενός νευρωνικού δικτύου, όπως θα αναλυθεί παρακάτω.

4.3 Εκπαίδευση ενός Νευρωνικού Δικτύου

Με τον όρο *εκπαίδευση* του νευρωνικού δικτύου εννοούμε την *ρύθμιση* ή *προσαρμογή* των ειδικών *παραμέτρων* του δικτύου, που ονομάζονται *βάρη* (weights) και *μεροληψίες* (στο εξής bias). Οι παράμετροι αυτοί αρχικοποιούνται τυχαία και με βάση τον αλγόριθμο που θα περιγραφεί παρακάτω προσπαθούμε να βρούμε τις βέλτιστες δυνατές τιμές αυτών.

Ας υποθέσουμε ότι έχουμε ένα νευρωνικό δίκτυο με N κρυφά στρώματα, δηλαδή συνολικά το δίκτυο έχει $N + 2$ στρώματα. Σε κάθε στρώμα αντιστοιχίζονται δύο *πίνακες*, ο πίνακας των βαρών, $\mathbf{W}^l_{jk}$ και ο πίνακας των biases, $\mathbf{b}^l_j$. Με $\mathbf{W}^l_{jk}$ συμβολίζουμε το βάρος για την σύνδεση του k-οστού νευρώνα του (l-1)-οστού στρώματος στον j-οστό νευρώνα του l-οστού στρώματος . Ως σύμβαση το στρώμα εισόδου θα έχει $l = 0$, τα κρυφά στρώματα θα έχουν $l = 1, \ldots, N$ ενώ το στρώμα εξόδου θα έχει $l = L$. Κάθε στοιχείο του πίνακα των βαρών μεταφράζεται ως η συνεισφορά ενός συγκεκριμένου νευρώνα στο υπολογισμό της εξόδου του επόμενου νευρώνα με τον οποίο συνδέεται. Το μοναδικό στρώμα για το οποίο δεν έχει νόημα να αντιστοιχίσουμε πίνακα βαρών είναι το στρώμα εξόδου αφού η προώθηση τελειώνει εκεί. Με $\mathbf{b}^l_j$ συμβολίζουμε το bias του j-οστού νευρώνα στο l-οστό στρώμα. Η διαστάσεις του πίνακα των βαρών και των biases καθορίζονται από την διάσταση (τον αριθμό των νευρώνων) των στρωμάτων στα οποία αναφέρεται. Ένας πίνακας με biases αντιστοιχίζεται σε όλα τα στρώματα εκτός από το στρώμα εισόδου και έχει διάσταση $1 \times n$, όπου n ο αριθμός των νευρώνων του αντίστοιχου στρώματος.

Για παράδειγμα ας υποθέσουμε ότι το στρώμα εισόδου ενός νευρωνικού δικτύου έχει 5 νευρώνες και το αμέσως επόμενο στρώμα, δηλαδή το πρώτο κρυφό στρώμα, έχει 10 νευρώνες (νούμερα πάρα πολύ μικρά για ένα τυπικό νευρωνικό δίκτυο!). Τότε ο πίνακας $\mathbf{W}^1_{jk}$ είναι ο πίνακας των βαρών που δίνει την συνεισφορά των στοιχείων του στρώματος εισόδου στον υπολογισμό της εξόδου των στοιχείων του πρώτου κρυφού στρώματος. Η διάσταση του πίνακα αυτού είναι $5 \times 10 = 50$, περιέχει δηλαδή 50 βάρη. Άρα προς το παρόν έχουμε 50

μεταβλητές. Επίσης για το πρώτο κρυφό στρώμα έχουμε επίσης 10 biases, άρα συνολικά 60 μεταβλητές. Τέλος ας υποθέσουμε ότι το νευρωνικό μας δίκτυο έχει μόνο ένα κρυφό στρώμα και το στρώμα εξόδου έχει 3 νευρώνες. Άρα ο πίνακας $\mathbf{W}^L_{jk}$ θα έχει 30 βάρη και επίσης στο στρώμα εξόδου θα αντιστοιχισθούν 3 βάρη. Συνολικά δηλαδή το νευρωνικό μας δίκτυο αποτελεί μία συνάρτηση $60 + 30 + 3 = 93$ μεταβλητών, οι οποίες όχι μόνο πρέπει να υπολογισθούν αλλά να βρεθεί και η βέλτιστη τιμή αυτών. Όπως είναι προφανές, ακόμα και για ένα πολύ απλό νευρωνικό δίκτυο, η εκπαίδευση αποτελεί μία πολύ περίπλοκη διαδικασία βελτιστοποίησης μιας συνάρτησης πάρα πολλών μεταβλητών.

Περιληπτικά η μεθοδολογία της εκπαίδευσης είναι η εξής: Η πληροφορία των δεδομένων εισόδου *προωθείται* μέσα στο νευρωνικό δίκτυο και επεξεργάζεται μέσω της διαδικασίας *feed forward*. Μόλις η πληροφορία φθάσει στο στρώμα εξόδου συγκρίνεται με τις αντίστοιχες επιθυμητές τιμές (που μπορεί να είναι για παράδειγμα πειραματικά δεδομένα ή δικές μας προβλέψεις). Η σύγκριση αυτή γίνεται με την βοήθεια μιας συνάρτησης κόστους. Πιο συγκεκριμένα λαμβάνεται η *μερική παράγωγος* της συνάρτησης κόστους ως προς όλες τις παραμέτρους του στρώματος εξόδου (βάρη και biases). Ανάλογα με το πρόσημο της κάθε μερικής παραγώγου το νευρωνικό δίκτυο καταλαβαίνει με ποιον τρόπο θα πρέπει να αλλάξει η κάθε παράμετρος *προκειμένου να ελαχιστοποιηθεί η συνάρτηση κόστους* που έχουμε επιβάλλει. Στην συνέχεια η πληροφορία αυτή *διαδίδεται προς τα πίσω* μέσω της διαδικασίας της *Προς-τα-πίσω διάδοσης* (Back-propagation). Η διαδικασία αυτή αποτελεί μία συνεχόμενη εφαρμογή του *κανόνα της αλυσίδας* και βοηθά το νευρωνικό δίκτυο να καταλάβει με ποιον τρόπο πρέπει να αλλάξουν όλες οι παράμετροι όλων των υπόλοιπων στρωμάτων του δικτύου. Μία επανάληψη του κύκλου που αποτελείται από μία διαδικασία feed-forward και μία διαδικασία back-propagation καλείται *εποχή* (epoche). Το νευρωνικό δίκτυο διανύει όσες εποχές χρειάζεται έτσι ώστε να βρεθεί ένα ελάχιστο (το οποίο τις περισσότερες φορές είναι τοπικό και πάρα πολύ σπάνια ολικό) της συνάρτησεις κόστους.

Συνεχίζουμε παρουσιάζοντας το πρώτο στάδιο της εκπαίδευσης ενός νευρωνικού δικτύου, την προώθηση της πληροφορίας.

4.4 Προώθηση (Feed Forward)

Το πρώτο στάδιο της εκπαίδευσης ενός νευρωνικού δικτύου είναι η εισαγωγή των δεδομένων μας σε αυτό, μέσω του στρώματος εισόδου, και η επεξεργασία τους από τα κρυφά στρώματα. Η διαδικασία είναι η εξής [8]:
Ένα δεδομένο $\mathbf{x}_k$ που εξέρχεται από έναν νευρώνα του στρώματος εισόδου πολλαπλασιάζεται με το στοιχείο $\mathbf{W}^1_{jk}$ και στο αποτέλεσμα προστίθεται το αντίστοιχο βάρος $\mathbf{b}^1_k$. Σχηματίζεται έτσι η ποσότητα:

$$\psi^1_{jk} = \mathbf{W}^1_{jk}\mathbf{x}_k + \mathbf{b}^1_k$$

$\forall k = 1, 2, \ldots, N$ όπου N ο αριθμός των νευρώνων του στρώματος εισόδου. Στην συνέχεια αθροίζουμε πάνω στον δείκτη k και παίρνουμε έτσι ένα *σταθμισμένο άθροισμα* των δεδομένων εισόδου. Το άθροισμα αυτό θα αποτελέσει την είσοδο του j νευρώνα του πρώτου

κρυφού στρώματος. Όπως προαναφέρθηκε, κάθε νευρώνας ενός κρυφού στρώματος αντιπροσωπεύει μία συνάρτηση ενεργοποίησης. Συνεπώς η έξοδος του j νευρώνα του πρώτου κρυφού στρώματος θα είναι το κανονικοποιημένο σταθμισμένο άθροισμα:

$$\alpha_j^1 = \sigma\left(\sum_k \psi_{jk}^1\right) = \sigma\left(\sum_k \mathbf{W}_{jk}^1 \mathbf{x}_k + \mathbf{b}_k^1\right) = \sigma\left(z_j^1\right) \tag{4.4.1}$$

Από το σημείο αυτό και μέχρι το στρώμα εξόδου η διαδικασία συνεχίζεται με παρόμοιο τρόπο. Στο πρώτο κρυφό στρώμα θα αντιστοιχισθούν οι πίνακες βαρών και biases. Για κάθε νευρώνα του πρώτου κρυφού στρώματος θα σχηματισθεί το σταθμισμένο άθροισμα και τελικά το άθροισμα αυτό θα περάσει σε όλους του νευρώνες του δεύτερου κρυφού στρώματος, όπου και θα δράσει η αντίστοιχη συνάρτηση ενεργοποίησης. Η έξοδος του κάθε νευρώνα του τελευταίου κρυφού στρώματος θα πολλαπλασιασθεί με το αντίστοιχο στοιχείο του τελευταίου πίνακα βαρών, θα προστεθεί το τελευταίο διάνυσμα biases και το τελικό αυτό αποτέλεσμα θα αποτελέσει το στρώμα εξόδου. Παίρνουμε έτσι τον αναδρομικό τύπο των τιμών ενεργοποίησης

$$\alpha_j^l = \sigma(z_j^l) = \sigma\left(\sum_k W_{jk}^l \alpha_k^{l-1} + b_j^l\right) \tag{4.4.2}$$

Η ποσότητα α_j^l ονομάζεται *τιμή ενεργοποίησης* (activation value). Το παραπάνω άθροισμα είναι πάνω σε όλους τους νευρώνες του $(l-1)$-οστού στρώματος.

Συνεχίζουμε την διερεύνηση της εκπαίδευσης ενός νευρωνικού δικτύου με μία μικρή παράκαμψη. Όπως αναφέρθηκε προηγουμένως τα δεδομένα του δικτύου επεξεργάζονται με την βοήθεια των λεγόμενων συναρτήσεων ενεργοποίησης. Θα εξετάσουμε τις πιθανές υποψήφιες κλάσσεις τέτοιων συναρτήσεων, ποιες είναι κατάλληλες και ποιες όχι.

4.5 Συναρτήσεις Ενεργοποίησης

Οι συναρτήσεις ενεργοποίησης είναι ο τρόπος με τον οποίο η πληροφορία των σημάτων εισόδου περνούν από τον έναν νευρώνα στον επόμενο. Σηματοδοτούν δηλαδή πότε ένας νευρώνας θα "ενεργοποιηθεί" και θα μεταβιβάσει την εκάστοτε πληροφορία. Υπάρχουν πολλές υποψήφιες επιλογές συναρτήσεων ενεργοποίησης και η επιλογή της "καταλληλότερης" συνάρτησης για τον κάθε νευρώνα (η συνηθέστερα για το κάθε κρυφό στρώμα) είναι ως επί το πλείστον εμπειρική και αποτελεί πηγή πολλών ερευνητικών δυνατοτήτων. Για να καταλάβουμε τί είδους συναρτήσεις είναι κατάλληλες για την μεταφορά πληροφορίας μεταξύ των νευρώνων μπορούμε να ξεκινήσουμε μελετώντας ποιες συναρτήσεις *δεν είναι κατάλληλες* για αυτόν τον σκοπό.

Η μελέτη των συναρτήσεων ενεργοποίησης ξεκίνησε με την εξής απλή σκέψη: Εφόσον ένας νευρώνας είτε θα μεταβιβάσει είτε δεν θα μεταβιβάσει την πληροφορία στον επόμενο νευρώνα, τι πιο φυσικό από το να χρησιμοποιήσουμε την *βηματική συνάρτηση* (step function):

$$\sigma(x) = \begin{cases} 1 & , x > 0 \\ 0 & , x < 0 \end{cases}$$

Παρόλο που η παραπάνω συνάρτηση φαίνεται διαισθητικά σωστή, στην πραγματικότητα εμφανίζει πολύ άσχημα αποτελέσματα στην πράξη της εκπαίδευσης των δικτύων. Αυτό οφείλεται στο γεγονός ότι ο ορισμός της παραπάνω συνάρτησης θα επιτρέψει στον νευρώνα είτε να μεταδώσει πλήρως την πληροφορία του είτε καθόλου, ενεργεί δηλαδή με δυαδικό τρόπο. Στην πράξη παρόλα αυτά μία καλή εκπαίδευση θα πρέπει να περιλαμβάνει και "ποσοστά μετάδοσης" πληροφορίας. Με αυτόν τον τρόπο το δίκτυό μας έχει περισσότερους βαθμούς ελευθερίας για να προσαρμόσει τις προς προσδιορισμό παραμέτρους του και αυτό προσφέρει μεγαλύτερη ευρωστία στην αρχιτεκτονική του.

Συνεπώς θα πρέπει να επιλέξουμε μία συνάρτηση που να δίνει και ενδιάμεσες τιμές ενεργοποίησης. Η πρώτη σκέψη είναι μία γραμμική συνάρτηση:

$$\sigma(x) = \alpha x$$

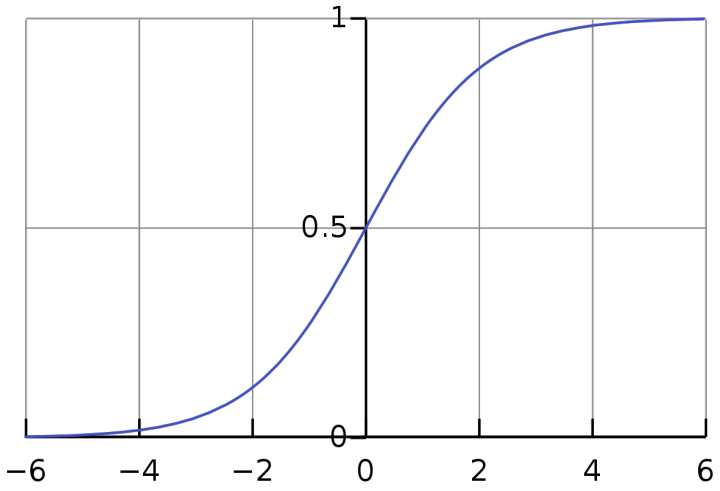
Μια συνάρτηση τέτοιας μορφής μπορεί να δώσει όντως μια μεγάλη ποικιλία τιμών ενεργοποίησης, οι οποίες τιμές είναι ανάλογες της εκάστοτε εισόδου. Παρόλα αυτά οι γραμμικές συναρτήσεις έχουν δύο σοβαρά μειονεκτήματα. Το πρώτο είναι το γεγονός ότι έχουν *σταθερή βαθμίδα* (ή *κλίση*). Αυτό σημαίνει ότι αν υπάρχει σφάλμα στην πρόβλεψη, δηλαδή η έξοδος του νευρωνικού δικτύου διαφέρει από την επιθυμητή τιμή, τότε η διόρθωση και οι αλλαγές που θα γίνονται με την προς-τα-πίσω μετάδοση θα είναι σταθερές και δεν θα εξαρτώνται από το σφάλμα αυτό. Το δεύτερο μειονέκτημα δεν είναι τόσο προφανές. Αν όλες οι συναρτήσεις ενεργοποίησης επιλεγούν γραμμικές τότε η τελική έξοδος του νευρωνικού δικτύου θα είναι και αυτή μία γραμμική συνάρτηση της εισόδου. Στην ουσία το σύνολο του νευρωνικού δικτύου (που μπορεί να περιέχει μέχρι και δεκάδες κρυφά στρώματα!) θα είναι ισοδύναμο με ένα νευρωνικό δίκτυο με ένα μόνο κρυφό στρώμα, εφόσον η σύνθεση γραμμικών συναρτήσεων είναι μία επίσης γραμμική συνάρτηση. Αυτό στην πράξη σημαίνει χειρότερη εκπαίδευση.

Το επόμενο βήμα δεν είναι ιδιαίτερα προφανές. Χρειαζόμαστε μία συνάρτηση που να έχει τουλάχιστον τις εξής ιδιότητες:

- Μη-γραμμική
- Φραγμένη (προτιμέστερα μεταξύ του 0 και του 1, χωρίς να έχει ιδιαίτερη σημασία)
- Να είναι τουλάχιστον μια φορά συνεχώς παραγωγίσιμη

Μία πολύ καλή υποψήφια συνάρτηση είναι η λεγόμενη *λογιστική συνάρτηση*:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



Εικόνα 3: Γραφική παράσταση της λογιστικής συνάρτησης

Η παραπάνω συνάρτηση έχει όλες τις παραπάνω επιθυμητές ιδιότητες συν άλλη μία η οποία δεν είναι άμεσα προφανής. Η λογιστική συνάρτηση έχει κλίση που δίνεται από τον τύπο:

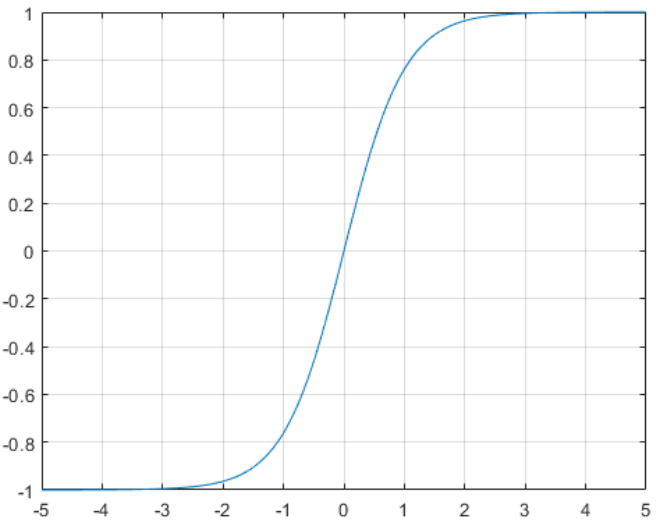
$$\nabla\sigma(x) = \frac{e^x}{(e^x+1)^2}$$

Στο διάστημα [-2,2] η κλίση της λογιστικής συνάρτησης είναι αρκετά μεγάλη, όπως φαίνεται και από την γραφική της παράσταση. Μεγάλη κλίση σημαίνει πως μικρές μεταβολές των μεταβλητών εισόδου (μικρά Δx) οδηγούν σε μεγάλες μεταβολές των μεταβλητών εξόδου. Υπάρχει δηλαδή αρκετά μεγάλη ευαισθησία σε μικρές διαταραχές του σήματος που εισέρχεται στον νευρώνα και αυτό συνεπάγεται πιο αποδοτική εκπαίδευση με την χρήση της προς-τα-πίσω μετάδοσης.

Η λογιστική συνάρτηση είναι όντως μία από τις ευρέως χρησιμοποιούμενες συναρτήσεις ενεργοποίησης στην τεχνολογία των νευρωνικών δικτύων λόγω των προαναφερθέντων ιδιοτήτων της. Στο σημείο αυτό όμως θα πρέπει να αναφερθεί και το μοναδικό μειονέκτημα της εν λόγω συνάρτησης που οφείλεται στην ίδια την φύση της. Παρατηρούμε από την παραπάνω γραφική παράσταση ότι στο σύνολο $[-\infty,-2]\cup[2,+\infty]$ η κλίση της λογιστικής συνάρτησης πρακτικά μηδενίζεται (η συνάρτηση τείνει να γίνει οριζόντια). Παρουσιάζεται δηλαδή το φαινόμενο των *εξαφανισμένων βαθμίδων* (vanishing gradients). Πολύ μικρές βαθμίδες στα διαστήματα αυτά σημαίνει πως μόλις η εκπαίδευση μας οδηγήσει στα "άκρα" της συνάρτησης τότε η εκπαίδευση επιβραδύνεται με πολύ μεγάλο ρυθμό και πρακτικά σταματά.

Το επόμενο βήμα αποτελεί μία βελτίωση της λογιστικής συνάρτησης και ονομάζεται *υπερβολική εφαπτομένη*:

$$\sigma(x) = \tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Εικόνα 4: Γραφική παράσταση της υπερβολικής εφαπτομένης

Συγκρίνοντας τις κλίσεις της λογιστικής συνάρτησης και της υπερβολικής εφαπτομένης παρατηρούμε ότι στο κοινό διάστημα [-2,2] η υπερβολική εφαπτομένη έχει μεγαλύτερη κλίση. Κατά τ'άλλα παρουσιάζει τα ίδια μειονεκτήματα και πλεονεκτήματα με την λογιστική συνάρτηση. Η επιλογή ανάμεσα στις δύο αυτές συναρτήσεις εξαρτάται από το πόσο μεγάλες κλίσεις θέλουμε να επιβάλλουμε στο νευρωνικό μας δίκτυο. Αποτελεί προς το παρόν θέμα κυρίως εμπειρικό και εξαρτάται από το εκάστοτε πρόβλημα.

Εναλλακτικές επιλογές συναρτήσεων που παρουσιάζουν παρόμοια χαρακτηριστικά με την λογιστική συνάρτηση και την υπερβολική εφαπτομένη είναι οι εξής συναρτήσεις:

$$f(x) = \arctan x$$
$$f(x) = \frac{x}{1+|x|}$$
$$f(x) = \frac{x}{\sqrt{1+\alpha x^2}}$$

Όλες οι παραπάνω μη-γραμμικές συναρτήσεις ανήκουν στην ειδικότερη κατηγορία των *σιγμοειδών* συναρτήσεων (sigmoid functions) και είναι η πιο ευρέως χρησιμοποιούμενη κατηγορία συναρτήσεων ενεργοποίησης στα νευρωνικά δίκτυα λόγω των ιδιοτήτων τους. Τα τελευταία χρόνια όμως έχει τεθεί σε εφαρμογή μια συνάρτηση που δεν είναι σιγμοειδής και ονομάζεται **ReLU** (Rectified Linear Unit). Δίνεται από την σχέση:

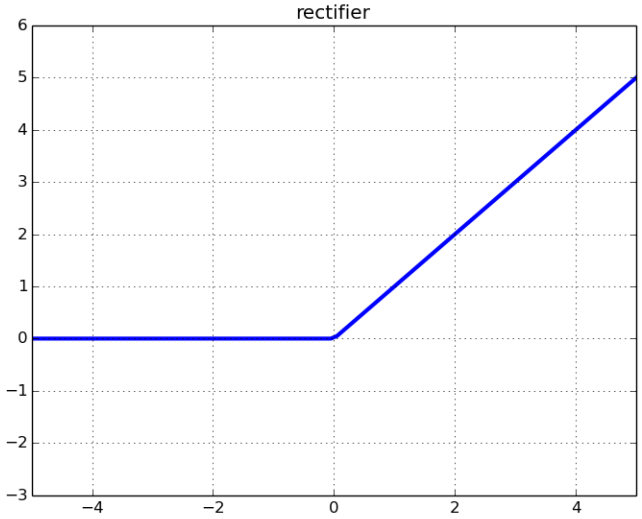
$$R(x) = \begin{cases} x & , x > 0 \\ 0 & , x < 0 \end{cases}$$

Η ReLU μπορεί να ορισθεί και ως $\mathrm{R(x)} = \max(0, x)$. Παρά το γεγονός ότι με μία πρώτη ματιά η ReLU φαίνεται γραμμική, στην πραγματικότητα δεν είναι. Μια γραμμική απεικόνιση

$T : \mathbb{R} \to \mathbb{R}$ ικανοποιεί τις εξής σχέσεις:

$$T(x + y) = T(x) + T(y)$$
$$T(\lambda x) = \lambda x$$

Είναι προφανές ότι η ReLU δεν ικανοποιεί την πρώτη συνθήκη των γραμμικών συναρτήσεων. Ως παράδειγμα έχουμε:
$R(1) = \max(0, 1) = 1$. Όμως $1 = 2 - 1$ και αν υποθέσουμε ότι η ReLU είναι γραμμική τότε θα ισχύει ότι: $R(2 - 1) = R(2) + R(-1) = 2 = R(1) = 1$ καί έτσι καταλήγουμε σε άτοπο και άρα στο συμπέρασμα ότι η ReLU δεν είναι γραμμική.



Εικόνα 5: Γραφική παράσταση της ReLU

Ένα ενδιαφέρον χαρακτηριστικό της ReLU είναι ότι μπορεί να δράσει πολύ καλά ως συναρτησιακός προσεγγιστής, με την έννοια ότι μπορούμε να προσεγγίσουμε όσο καλά θέλουμε μία συνεχή συνάρτηση με την χρήση γραμμικού συνδυασμού συναρτήσεων ReLU. Πιο φορμαλιστικά, για κάθε $\epsilon > 0$ και για κάθε συνεχή συνάρτηση f υπάρχει ένας φυσικός N, πραγματικοί αριθμοί α_i και συναρτήσεις ReLU ορισμένες σε κατάλληλο εκάστοτε διάστημα, ώστε :

$$\left| f - \sum_{i=1}^{N} \alpha_i R_i(x) \right| < \epsilon$$

Στις περισσότερες περιπτώσεις η ReLU χρησιμοποιείται σε συνδυασμό με άλλες συναρτήσεις ενεργοποίησης, συνήθως σιγμοειδείς. Η παρουσία της προσφέρει ένα σημαντικό πλεονέκτημα. Λόγω της μορφής της δεν επιτρέπει σε όλους τους νευρώνες να ενεργοποιηθούν. Αν φανταστούμε ένα νευρωνικό δίκτυο με εκατοντάδες ή και χιλιάδες νευρώνες τότε προφανώς ο υπολογιστικός χρόνος της εκπαίδευσης μπορεί να είναι πολύ μεγάλος για να είναι πρακτικός. Η χρήση της ReLU λοιπόν μας βοηθά να δημιουργήσουμε ένα νευρωνικό δίκτυο

πιο αραιό και ανάλαφρο και έτσι να εξοικονομήσουμε πολύτιμο υπολογιστικό χρόνο. Αυτό είναι το μεγάλο πλεονέκτημά της. Το πρόβλημα που δημιουργεί είναι προφανές στο σημείο αυτό και είναι η μηδενική κλίση (δηλαδή τερματισμός της εκπαίδευσης) για αρνητικές τιμές εισόδου. Το πρόβλημα αυτό λύνεται με μικρές παραλλαγές της ReLU που παίρνουν την μορφή:

$$R_\epsilon(x) = \begin{cases} x & , x > 0 \\ \epsilon x & , x < 0 \end{cases}$$

όπου $\epsilon << 1$. Η μικρή παράμετρος ϵ μπαίνει έτσι ώστε να υπάρχει μη-μηδενική κλίση για αρνητικές τιμές εισόδου. Τέτοιες παραλλαγές της ReLU είναι γνωστές ως **Leaky ReLU**.

4.6 Συναρτήσεις Κόστους

Η ανάγκη για την ύπαρξη μίας συνάρτησης κόστους στο νευρωνικό δίκτυο έγκειται στο γεγονός ότι χρειαζόμαστε ένα κριτήριο με βάση το οποίο θα γίνει η εκπαίδευση του δικτύου. Οι συναρτήσεις κόστους μας βοηθούν να συγκρίνουμε τις τιμές του στρώματος εξόδου με τις επιθυμητές τιμές (προβλέψεις ή πειραματικά δεδομένα). Είναι δηλαδή ο τρόπος με τον οποίο βοηθούμε τον υπολογιστή να καταλάβει αν η διαδικασία της προώθησης ήταν επιτυχής. Για λόγους που δεν είναι άμεσα προφανείς και σχετίζονται με την διαδικασία του back-propagation οι συναρτήσεις κόστους πρέπει να πληρούν τις δύο εξής προϋποθέσεις [9]:

- Η συνάρτηση κόστους θα πρέπει να μπορεί να γραφεί ως ένας μέσος όρος $C = \frac{1}{n}\sum_x C_x$ συναρτήσεων κόστους για κάθε μεμονωμένο αντικείμενο εκπαίδευσης, x. Ο λόγος που χρειαζόμαστε αυτήν την προϋπόθεση είναι επειδή στον αλγόριθμο back-propagation απαιτείται ο υπολογισμός μερικών παραγώγων ως προς όλα τα αντικείμενα προς εκπαίδευση.

- Η συνάρτηση κόστους θα πρέπει να μπορεί να γραφτεί συναρτήσει των εξόδων του νευρωνικού δικτύου $C = C(\alpha^l)$

Μερικές από τις ευρέως χρησιμοποιούμενες συναρτήσεις κόστους είναι οι εξής:

Συνάρτηση μέσου τετραγωνικού σφάλματος (Mean Square Error Function):

$$C = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

όπου y_i οι τιμές του στρώματος εξόδου του δικτύου και $\hat{y}_i$ οι προβλέψεις μας με τις οποίες τις συγκρίνουμε. Η συνάρτηση αυτή χρησιμοποιείται ευρέως σε προβλήματα παλινδρόμησης (regression) όμως έχει ένα σοβαρό πρόβλημα. Σε περίπτωση που υπάρχουν δεδομένα (παρατηρούμενες τιμές) που απέχουν πολύ από όλα τα υπόλοιπα, τα λεγόμενα outliers, τότε η εφαρμογή της συνάρτησης τετραγωνικού σφάλματος μπορεί να έχει πολύ μεγάλα σφάλματα

στους υπολογισμούς. Για τον σκοπό αυτό επινοήθηκε η συνάρτηση σφάλματος κατά Huber.

Συνάρτηση Huber (Huber Loss):

$$C = \begin{cases} \frac{1}{2}\sum_i (y_i - \hat{y}_i)^2 & , |y_i - \hat{y}_i| \leq \delta \ \ \forall i \\ \delta \sum_i |y_i - \hat{y}_i| - \frac{1}{2}\delta^2 & , \alpha\lambda\lambda\iota\omega\varsigma \end{cases}$$

Η συνάρτηση Huber είναι πολύ λιγότερο ευαίσθητη σε outliers.

Συνάρτηση Cross Entropy:

$$C = -\sum_{i=1}^{n} y_i \log \hat{y}_i$$

Η Cross Entropy αποτελεί μια από τις σημαντικότερες συναρτήσεις για το κλάδο τον Μαθηματικών με όνομα Θεωρία Πληροφορίας.

Όλες οι μορφές των παραπάνω συναρτήσεων αφορούν φυσικά διακριτό χώρο δειγματοληψίας. Αν οι κατανομές μας αποτελούν συνεχείς συναρτήσεις τότε το άθροισμα αντικαθιστάται με ολοκλήρωμα.

4.7 Κανονικοποίηση Κόστους

Η κανονικοποίηση κόστους (ή κανονικοποίηση συνάρτησης κόστους) είναι μία τεχνική που χρησιμοποιούνταν αρκετές δεκαετίες πριν την ανάπτυξη της εκπαίδευσης νευρωνικών δικτύων βάθους. Γραμμικά μοντέλα όπως η γραμμική και η λογιστική παρεμβολή δίνουν την ικανότητα εφαρμογής απλών και αποτελεσματικών τεχνικών κανονικοποίησης.

Οι εφαρμογές κανονικοποίησης βασίζονται στον περιορισμό των ικανοτήτων και της χωρητικότητας του μοντέλου (είτε αυτό είναι νευρωνικά δίκτυα ή κάποιου είδους παρεμβολή) προσθέτοντας μια παραμετρικοποιημένη ποινή $\Omega(\boldsymbol{\vartheta})$ στην αντικειμενική συνάρτηση κόστους $\mathcal{J}$. Συμβολίζουμε την κανονικοποιημένη συνάρτηση κόστους ώς

$$\widetilde{\mathcal{J}} = \mathcal{J}(\boldsymbol{\vartheta}; \mathbf{x}, \mathbf{y}) + \alpha\Omega(\boldsymbol{\vartheta}) \tag{4.7.1}$$

όπου η πολλαπλασιαστική σταθερά $\alpha \in [0, \infty)$ είναι μία υπερπαράμετρος που σταθμίζει την σχετική συνεισφορά του όρου της ποινής, Ω, στην συνάρτηση κόστους, $\mathcal{J}$. Φυσικά θέτοντας $\alpha = 0$ έχει ως αποτέλεσμα να μην έχουμε καμία κανονικοποίηση. Όσο το α αυξάνεται τόσο αυξάνεται και η συνεισφορά της ποινής στην αντικειμενική συνάρτηση. Όταν ο αλγόριθμος εκπαίδευσής μας προσπαθεί να ελαχιστοποιήσει την αντικειμενική συνάρτηση $\widetilde{\mathcal{J}}$, τότε θα μειώνει και την συνάρτηση $\mathcal{J}$ αλλά και μία συνάρτηση της οποίας το μέτρο θα

εξαρτάται καθαρά και μόνο από τις παραμέτρους $\boldsymbol{\vartheta}$. Διαφορετικές επιλογές της συνάρτησεις Ω θα έχουν φυσικά διαφορετική επίδραση στην εκπαίδευση. Στο σημείο αυτό αξίζει να σημειώσουμε ότι στις περισσότερες εφαρμογές των νευρωνικών δικτύων επιβάλλουμε ποινή μόνο στα βάρη του δικτύου και όχι στα biases [10]. Κάθε βάρος προσδιορίζει τον τρόπο που δύο μεταβλητές αλληλεπιδρούν. Η προσαρμογή ενός βάρους λοιπόν απαιτεί την παρατήρηση και των δύο αυτών μεταβλητών σε πολλές περιστάσεις. Εν αντιθέσει, ένα bias ελέγχει μόνο μία μεταβλητή. Αυτό σημαίνει ότι δεν εισάγουμε πολύ διακύμανση αφήνοντας τα βάρη μη-κανονικοποιημένα. Για το υποκεφάλαιο αυτό θα συμβολίζουμε με $\mathbf{w}$ το διάνυσμα των βαρών ενός νευρωνικού δικτύου στα οποία θα επιβάλλεται μία ποινή κανονικοποίησης και με $\boldsymbol{\vartheta}$ θα συμβολίζουμε όλες τις παραμέτρους του δικτύου (όλα τα βάρη και όλα τα biases). Συνεχίζουμε με τα διάφορα είδη κανονικοποίησης ποινής.

- $\mathcal{L}^2$ - Κανονικοποίηση ή Κανονικοποίηση *Tykhonov*

Πρόκεται δηλαδή για την $\mathcal{L}^2$ νόρμα του διανύσματος των βαρών στα οποία επιβάλλουμε την επιπλέον ποινή. Η ποινή κόστους τύπου $\mathcal{L}^2$ είναι η πιο κοινή στην πράξη. Βοηθά στο να κρατά τα βάρη πιο κοντά στο 0 [11], όπως θα δείξει η μαθηματική ανάλυση που θα ακολουθήσει. Η συνάρτηση κανονικοποίησης Ω δίνεται από την σχέση

$$\Omega(\boldsymbol{\vartheta}) = \frac{1}{2}\|\mathbf{w}\|_2^2 = \frac{1}{2}\mathbf{w}^\intercal\mathbf{w} \tag{4.7.2}$$

Με βάση την παραπάνω ποινή η αντικειμενική συνάρτηση γράφεται ως

$$\widetilde{\mathcal{J}} = \mathcal{J}(\boldsymbol{\vartheta}; \mathbf{x}, \mathbf{y}) + \frac{\alpha}{2}\mathbf{w}^\intercal\mathbf{w} \tag{4.7.3}$$

Η συνάρτηση αυτή έχει βαθμίδα ως προς τα βάρη

$$\nabla_{\mathbf{w}}\widetilde{\mathcal{J}} = \alpha\mathbf{w} + \nabla_{\mathbf{w}}\mathcal{J}(\boldsymbol{\vartheta}; \mathbf{x}, \mathbf{y})$$

Με βάση αυτήν την βαθμίδα ένα βήμα του αλγορίθμου gradient descent επικαιροποιεί το διάνυσμα των παραμέτρων $\mathbf{w}$ ως εξής

$$\mathbf{w} \leftarrow \mathbf{w} - \tau(\alpha\mathbf{w} + \nabla_{\mathbf{w}}\mathcal{J}(\boldsymbol{\vartheta}; \mathbf{x}, \mathbf{y}))$$

ή, αν αναδιατάξουμε τους όρους

$$\mathbf{w} \leftarrow (1 - \tau\alpha)\mathbf{w} - \tau\nabla_{\mathbf{w}}\mathcal{J}(\boldsymbol{\vartheta}; \mathbf{x}, \mathbf{y}) \tag{4.7.4}$$

Όπως φαίνεται από την παραπάνω εξίσωση η εισαγωγή της ποινής $\mathcal{L}^2$ τροποποίησε τον κανόνα επικαιροποίησης. Σε κάθε επανάληψη της διαδικασίας το διάνυσμα των βαρών συρρικνώνεται κατά έναν σταθερό παράγοντα. Στην συνέχεια θα εξετάσουμε την επίδραση της $\mathcal{L}^2$ ποινής κατά την διάρκεια όλης της εκπαίδευσης. Προς απλοποίηση των πράξεων του συλλογισμού που θα ακολουθήσει θα κάνουμε μία τετραγωνική προσέγγιση (στην ουσία

ανάπτυγμα Taylor δευτέρου βαθμού) της μη-κανονικοποιημένης αντικειμενικής συνάρτησης $\mathcal{J}$ γύρω από το σημείο που την ελαχιστοποιεί, $\mathbf{w}^* = \text{argmin}_{\mathbf{w}} \mathcal{J}(\mathbf{w})$. Γράφουμε δηλαδή

$$\hat{\mathcal{J}}(\mathbf{w}) = \mathcal{J}(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^\intercal \mathbf{H}(\mathbf{w} - \mathbf{w}^*)$$

όπου $\mathbf{H}$ είναι ο Εσσιανός πίνακας της $\mathcal{J}$ ως προς το διάνυσμα $\mathbf{w}$ υπολογισμένος στο σημείο $\mathbf{w}^*$. Προφανώς αν η συνάρτηση κόστους είναι τετραγωνική (που στις περισσότερες εφαρμογές είναι) τότε η προσέγγιση συμπίπτει με την ίδια την συνάρτηση. Το ελάχιστο της συνάρτησης $\hat{\mathcal{J}}(\mathbf{w})$ λαμβάνεται εκεί όπου η βαθμίδα

$$\nabla_{\mathbf{w}} \hat{\mathcal{J}}(\mathbf{w}) = \mathbf{H}(\mathbf{w} - \mathbf{w}^*)$$

είναι ίση με 0. Θα συμβολίσουμε το σημειό στο οποίο ικανοποιείται η παραπάνω συνθήκη με $\hat{\mathbf{w}}$. Για να εξετάσουμε την επίδραση της ποινής $\mathcal{L}^2$ θα πρέπει να προσθέσουμε στην παραπάνω συνθήκη και την βαθμίδα του όρου της ποινής, $\alpha\hat{\mathbf{w}}$. Λαμβάνουμε έτσι

$$\begin{aligned} \alpha\hat{\mathbf{w}} + \mathbf{H}(\mathbf{w} - \mathbf{w}^*) \\ (\mathbf{H} + \alpha\mathbf{I})\hat{\mathbf{w}} = \mathbf{H}\mathbf{w}^* \\ \hat{\mathbf{w}} = (\mathbf{H} + \alpha\mathbf{I})^{-1}\mathbf{H}\mathbf{w}^* \end{aligned}$$

Παρατηρούμε ότι καθώς $\alpha \to 0$, $\hat{\mathbf{w}} \to \mathbf{w}^*$. Επειδή ο πίνακας $\mathbf{H}$ είναι πραγματικός και συμμετρικός μπορούμε να τον γράψουμε ως γινόμενο ενός διαγώνιου πίνακα $\mathbf{\Lambda}$ και έναν πίνακα $\mathbf{Q}$ που αποτελείται από μία ορθοκανονική βάση ιδιοδιανυσμάτων, ώστε $\mathbf{H} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\intercal$ [10]. Με βάση αυτή την αποσύνθεση λαμβάνουμε

$$\begin{aligned} \hat{\mathbf{w}} &= (\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\intercal + \alpha\mathbf{I})^{-1}\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\intercal\mathbf{w}^* \\ &= (\mathbf{Q}(\mathbf{\Lambda} + \alpha\mathbf{I})\mathbf{Q}^\intercal)^{-1}\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\intercal\mathbf{w}^* \\ &= \mathbf{Q}(\mathbf{\Lambda} + \alpha\mathbf{I})^{-1}\mathbf{\Lambda}\mathbf{Q}^\intercal\mathbf{w}^* \end{aligned}$$

Παραρηρούμε ότι η επίδραση της ποινής $\mathcal{L}^2$ είναι ακριβώς ότι ανακλιμακώνει (rescales) το διάνυσμα $\mathbf{w}^*$ κατά μήκος των αξόνων που ορίζονται από τα ιδιοδιανύσματα του πίνακα $\mathbf{H}$.

Συνεχίζουμε τώρα με την ποινή τύπου $\mathcal{L}^1$.

- $\mathcal{L}^1 - Κανονικοποιηση$

Στην περίπτωση της κανονικοποίησης τύπου $\mathcal{L}^1$ η συνάρτηση Ω δίνεται από την σχέση

$$\Omega(\boldsymbol{\vartheta}) = \|\mathbf{w}\|_1 = \sum_i |w_i| \tag{4.7.5}$$

Πρόκειται δηλαδή για την $\mathcal{L}^1$ νόρμα του διανύσματος των βαρών στα οποία επιβάλλουμε την επιπλέον ποινή. Τώρα η κανονικοποιημένη αντικειμενική συνάρτηση έχει την μορφή

$$\widetilde{\mathcal{J}}(\mathbf{w}; \mathbf{x}, \mathbf{y}) = \alpha\|\mathbf{w}\|_1 + \mathcal{J}(\mathbf{w}; \mathbf{x}, \mathbf{y}) \tag{4.7.6}$$

όπου, ομοίως με πριν, η σταθερά αείναι μία υπερπαράμετρος που καθορίζει το μέγεθος της επίδρασης της κανονικοποίησης. Η βαθμίδα της κανονικοποιημένης αντικειμενικής συνάρτησης είναι

$$\nabla_{\mathbf{w}}\widetilde{\mathcal{J}}(\mathbf{w};\mathbf{x},\mathbf{y}) = \alpha sign(\mathbf{w}) + \nabla_{\mathbf{w}}\mathcal{J}(\mathbf{w};\mathbf{x},\mathbf{y}) \tag{4.7.7}$$

Από την παραπάνω βαθμίδα βλέπουμε ότι η επίδραση της $\mathcal{L}^1$ κανονικοποίησης είναι διαφορετική από αυτήν της $\mathcal{L}^2$. Πιο συγκεκριμένα η συνεισφορά της κανονικοποίησης στην βαθμίδα δεν αυξομειώνεται γραμμικά με την κάθε συνιστώσα w_i. Είναι απλώς μία πολλαπλασιαστική σταθερά αμε πρόσημο που εξαρτάται από το $sign(w_i)$, δηλαδή από το πρόσημο της συνιστώσας w_i. Μία συνέπεια αυτής της μορφής βαθμίδας είναι ότι μία τετραγωνική προσέγγιση της αντικειμενικής συνάρτησης δεν θα μπορεί να δώσει απαραίτητα κάποια αναλυτική λύση όπως στην περίπτωση της κανονικοποίησης $\mathcal{L}^2$. Ο κανόνας επικαιροποίησης του διανύσματος των βαρών θα είναι τώρα

$$\mathbf{w} \leftarrow \mathbf{w} + \tau(\alpha sign(\mathbf{w}) + \nabla_{\mathbf{w}}\mathcal{J}(\mathbf{w};\mathbf{x},\mathbf{y})) \tag{4.7.8}$$

4.8 Κατάβαση Βαθμίδας (Gradient Descent)

Η διαδικασία του Gradient Descent είναι ένας αλγόριθμος με τον οποίο μπορούμε να βρούμε ένα *τοπικό ακρότατο* μιας πολυμεταβλητής συνάρτησης. Η ιδέα πίσω από τον αλγόριθμο είναι απλή. Ξεκινώντας από ένα αυθαίρετο σημείο πάνω στην (υπερ)επιφάνεια που ορίζει η συνάρτηση μας τότε αν θέλουμε να κάνουμε ανάβαση της επιφάνειας, δηλαδή να πάμε προς το πιο κοντινό μέγιστο, τότε θα πρέπει να ακολουθήσουμε την κατεύθυνση που ορίζει η *βαθμίδα* (gradient) της συνάρτησης στο σημείο αυτό. Αυτό είναι και το φυσικό νόημα του gradient. Σε κάθε σημείο της επιφάνειας μας λέει προς τα που πρέπει να κινηθούμε έτσι ώστε να έχουμε την μεγαλύτερη αύξηση ή *κλίση*. Συνεπώς, αν επιθυμούμε να κάνουμε κατάβαση της επιφάνειας τότε θα πρέπει να ακολουθήσουμε πορεία *αντίθετη της βαθμίδας*. Σε ακριβώς αυτό έγκειται ο αλγόριθμος του gradient descent, δηλαδή στην επικαιροποίηση ενός τυχαίου σημείου της επιφάνειας με βήμα ανάλογο της βαθμίδας.

Ας υποθέσουμε λοιπόν ότι μας δίνεται ένα συνεκτικό ανοιχτό χωρίο $D \subset \mathbb{R}^d$ και μία πραγματική συνάρτηση $f \in C^1(D)$, η οποία θα αποκαλείται *αντικειμενική συνάρτηση* (objective function)

$$f : D \to \mathbb{R}$$

την οποία επιθυμούμε να ελαχιστοποιήσουμε. Ψάχνουμε δηλαδή ένα $\mathbf{x}^*$, τέτοιο ώστε:

$$\mathbf{x}^* = \underset{\mathbf{x}\in D}{\operatorname{argmin}} f(\mathbf{x})$$

Προφανώς επίσης θα ισχύει

$$\nabla f(\mathbf{x}^*) = 0$$

Ο αλγόριθμος απαιτεί να του ορίσουμε το αρχικό σημείο πάνω στην επιφάνεια από το οποίο θα αρχίσει η κατάβαση (ή η ανάβαση), το *βήμα επικαιροποίησης* t_k και ένα *όριο ανοχής*

(tolerance) το οποίο θα καθορίσει πότε θα σταματήσει ο αλγόριθμος. Συνοπτικά ο αλγόριθμος gradient descent φαίνεται στον παρακάτω πίνακα.

Αλγόριθμος Gradient Descent
1: Αρχικοποίηση τυχαίου σημείου $\mathbf{x}^{(0)}$, βήματος t_k και tolerance ϵ
2: **while** $\|\|\nabla f(\mathbf{x}^{(k)})\|\| \geq \epsilon$ **do** :
3: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - t_k \nabla f(\mathbf{x}^{(k)})$
4: $k \leftarrow k + 1$
5: **end while**
6: **return** $\mathbf{x}^{(k)}$

Πίνακας 1: Αλγόριθμος Gradient Descent

Μερικά σχόλια για τον αλγόριθμο και τις παραμέτρους του. Καταρχάς ο παραπάνω αλγόριθμος αναφέρεται σε προβλήματα βελτιστοποίησης *χωρίς περιορισμούς*, δηλαδή θεωρούμε ότι όλα τα σημεία του χώρου που ορίζονται από την επιφάνεια που ορίζει η αντικειμενική συνάρτηση είναι προσβάσιμα. Ο αλγόριθμος τροποποιείται για προβλήματα που περιέχουν περιορισμούς όμως έχει την ίδια επιτυχία.

Στο τρίτο βήμα του αλγορίθμου παρατηρούμε ότι χρησιμοποιούμε το πρόσημο "-" για να κάνουμε επικαιροποίηση του σημείου της επιφάνειας. Ο λόγος που επιλέγουμε αυτό το πρόσημο είναι ακριβώς επειδή θέλουμε να ακολουθήσουμε μία πορεία που είναι *αντίθετη* της βαθμίδας, με σκοπό να οδηγηθούμε στο ελάχιστο της συνάρτησης. Αν, αντιθέτως, χρησιμοποιούσαμε το πρόσημο "+" τότε θα ακολουθούσαμε την πορεία της ίδιας της βαθμίδας και θα πηγαίναμε προς το μέγιστο της συνάρτησης. Στην περίπτωση αυτή ο αλγόριθμος λέγεται *ανάβαση βαθμίδας* (Gradient Ascent).

Το βήμα επικαιροποίησης t_k μπορεί να είναι είτε ένας σταθερός αριθμός είτε να εξαρτάται από το βήμα στο οποίο βρισκόμαστε. Η δεύτερη περίπτωση είναι πιο αποδοτική αν το βήμα επιλεγεί με έναν σχετικά έξυπνο τρόπο. Το κόστος φυσικά είναι αρκετά μεγαλύτεροι υπολογιστικοί χρόνοι. Η λογική πίσω από την επιλογή του βήματος είναι η σκέψη ότι θέλουμε να ελαχιστοποιεί την αντικειμενική μας συνάρτηση σε μια περιοχή του βήματος στο οποίο πρόκειται να βρεθούμε μετά από το εκάστοτε βήμα. Σε μαθηματικούς όρους, το *βέλτιστο βήμα* t_k^* θα ικανοποιεί [12]:

$$t_k^* = \underset{t \geq 0}{\operatorname{argmin}} f(\mathbf{x}^{(k+1)}) = \underset{t \geq 0}{\operatorname{argmin}} f(\mathbf{x}^{(k)} - t \nabla f(\mathbf{x}^{(k)})$$

Φυσικά είναι εξαιρετικά λίγες οι περιπτώσεις όπου το βήμα αυτό μπορεί να υπολογισθεί αναλυτικά μιας και στις περισσότερες εφαρμογές οι αντικειμενικές συναρτήσεις είναι μέχρι και δεκάδων μεταβλητών.

Το όριο ϵ είναι το κριτήριο τερματισμού του αλγορίθμου και είναι ένας πραγματικός αριθμός πολύ μικρότερος της μονάδας και φυσικά θετικός. Όταν η νόρμα (δηλαδή το μέγεθος)

της βαθμίδας, $||\nabla f||$, πέσει κάτω από ϵ μετά την κ-οστή επανάληψη τότε αυτό σημαίνει ότι σε περιοχή του σημείου $\mathbf{x}^{(k)}$ η κλίση της συνάρτηση είναι σχεδόν μηδενική και άρα ως γνωστόν θα είμαστε πολύ κοντά στο ακρότατο. Θεωρητικά μπορούμε να προσεγγίσουμε το ακρότατο αυτό όσο καλά θέλουμε μικραίνοντας όλο και πιο πολύ το ϵ.

Φυσικά, επειδή καταφέραμε να βρούμε έναν τρόπο να προσεγγίσουμε ένα τοπικό ακρότατο μίας συνάρτησης δεν σημαίνει ότι μπορούμε πάντα να εντοπίσουμε το *ολικό ελάχιστο* μιας συνάρτησης. Στην πραγματικότητα, ο προσδιορισμός του ολικού ελαχίστου μια πολυμεταβλητής συνάρτησης είναι ένα υπέρμετρα δύσκολο μαθηματικό πρόβλημα. Για τον λόγο αυτό στις περισσότερες εφαρμογές η εύρεση ενός τοπικού ακροτάτου της αντικειμενικής μας συνάρτησης θεωρείται επιτυχία. Επίσης κρίνεται σκόπιμο να αναφερθεί ότι στην περίπτωση που η αντικειμενική μας συνάρτηση είναι κυρτή (αντ. κοίλη) τότε γνωρίζουμε από τον μαθηματικό λογισμό ότι θα παρουσιάζει μοναδικό ελάχιστο (αντ. μέγιστο). Συνεπώς στις περιπτώσεις αυτές ο αλγόριθμος gradient descent μας εγγυάται σύγκλιση αφού η βαθμίδα πάντα θα οδηγεί προς το μέγιστο ή το ελάχιστο

4.9 Σύγκλιση του αλγορίθμου Gradient Descent

Ένα εύλογο ερώτημα που προκύπτει είναι κατά πόσο ο αλγόριθμος gradient descent δουλεύει, δηλαδή κατά πόσο συγκλίνει. Η απάντηση δεν είναι απλή και χρειαζόμαστε, μεταξύ άλλων, τον παρακάτω ορισμό:

Ορισμός: Έστω ένα κυρτό σύνολο $D \subset \mathbb{R}^d$, μία κυρτή συνάρτηση $f : D \to \mathbb{R}$. Η f θα λέγεται **ισχυρώς κυρτή** (strongly convex) αν υπάρχει θετικός πραγματικός αριθμός m τέτοιος ώστε για κάθε $x, y \in D$:

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq m \parallel x - y \parallel^2$$

όπου $\| \cdot \|$ μία οποιαδήποτε νόρμα. Αποδεικνύεται [13] ότι η παραπάνω ανισότητα ισοδυναμεί με την ύπαρξη θετικών πραγματικών αριθμών m και M για τους οποίους ισχύει:

$$mI \leq \mathcal{H}_f(\mathbf{x}) \leq MI$$

όπου $\mathcal{H}_f(\mathbf{x})$ ο εσσιανός πίνακας της f στο $\mathbf{x}$ και I ο μοναδιαίος πίνακας. Σημειώνεται ότι για δύο πίνακες A και B ίδιων διαστάσεων $A \geq B$ σημαίνει ότι ο πίνακας A - B είναι θετικά ημι-ορισμένος. Μπορούμε έυκολα να δείξουμε ότι μία ισχυρώς κυρτή συνάρτηση είναι και αυστηρά κυρτή [13]. Παίρνοντας το ανάπτυγμα Taylor δεύτερης τάξης για την f έχουμε ότι για $x, y \in D$ μπορούμε να βρούμε ένα $z \in [x, y]$ τέτοιο ώστε:

$$f(y) = f(x) + \nabla f(x)^\intercal (y - x) + \frac{1}{2}(y - x)^\intercal \mathcal{H}_f (y - x)$$

και λόγω της παραπάνω διπλής ανισότητα παίρνουμε:

$$f(y) \geq f(x) + \nabla f(x)^\intercal (y - x) + \frac{m}{2} \parallel y - x \parallel^2$$

και επειδή $\frac{m}{2} \parallel y - x \parallel^2 > 0$:

$$f(y) > f(x) + \nabla f(x)^\intercal (y - x)$$

Έχοντας τώρα την ιδιότητα της ισχυρής κυρτότητας παίρνουμε το εξής θεώρημα [12]:

Θεώρημα: Έστω $f : D \to \mathbb{R}$ μία ισχυρώς κυρτή συνάρτηση με παραμέτρους m και M, όπως και στην παραπάνω διπλή ανισότητα, και $\alpha^* = \min_{\mathbf{x} \in D} f(\mathbf{x})$. Τότε για κάθε $\epsilon > 0$ μπορούμε να πετύχουμε $f(\mathbf{x}^{(k^*)}) - \alpha^* \leq \epsilon$ μετά από k^* επαναλήψεις για κάποιο κ^* που ικανοποιεί:

$$k^* \geq \frac{\log\left(\frac{f(\mathbf{x}^{(0)}) - \alpha^*}{\epsilon}\right)}{\log\left(\frac{1}{1 - \frac{m}{M}}\right)}$$

Το παραπάνω θεώρημα μας εξασφαλίζει ότι μετά από έναν αριθμό επαναλήψεων που εξαρτάται από την φύση της ίδιας της συνάρτησης (τις σταθερές m και M) αλλά και από την ακρίβεια που επιθυμούμε μπορούμε να φτάσουμε όσο κοντά θέλουμε στο ακρότατο της f. Προφανώς όταν $\epsilon \to 0$ τότε $k^* \to +\infty$. Επίσης παρατηρούμε ότι καθώς $m \to M$ απαιτούνται όλο και λιγότερες επαναλήψεις για να συγκλίνει ο αλγόριθμος.

Προφανώς ο ρόλος του αλγορίθμου του gradient descent στα πλαίσια των νευρωνικών δικτύων είναι η ελαχιστοποίηση των συναρτήσεων κόστους ως προς τις παραμέτρους του δικτύου, δηλαδή τα βάρη και τα biases. Συνεπώς ένα κρίσιμο βήμα της εκπαίδευσης είναι ο υπολογισμός των βαθμίδων της συνάρτησης κόστους ως προς όλες αυτές τις παραμέτρους ξεχωριστά. Στο σημείο αυτό όμως, έχοντας αρχίσει να εφαρμόζουμε τον αλγόριθμο gradient descent, παρατηρούμε ότι παρουσιάζεται ένα μείζονος σημασίας πρόβλημα και αυτό είναι οι τεράστιοι υπολογιστικοί χρόνοι που απαιτούνται για να τερματίσει ο αλγόριθμος. Όπως είδαμε και στην ενότητα **4.6** μία τυπική συνάρτηση κόστους έχει την γενική μορφή:

$$\mathbf{J}(\boldsymbol{\vartheta}) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{\vartheta})$$

όπου $\mathcal{L}$ μία συνάρτηση σφάλματος (π.χ. η διαφορά των τετραγώνων) και $\boldsymbol{\vartheta}$ το διάνυσμα των παραμέτρων ως προς τις οποίες επιθυμούμε να γίνει η βελτιστοποίηση, δηλαδή η εύρεση των σημείων μηδενισμού της βαθμίδας

$$\nabla_{\boldsymbol{\vartheta}} \mathbf{J}(\boldsymbol{\vartheta}) \ = \frac{1}{n} \sum_{i=1}^{n} \nabla_{\boldsymbol{\vartheta}} \mathcal{L}(\boldsymbol{\vartheta})$$

Ο υπολογιστικός χρόνος (ή υπολογιστικό κόστος) της παραπάνω πράξης είναι $\mathcal{O}(n)$. Συνεπώς όταν ο αριθμός των δεδομένων, n, γίνεται πάρα πολύ μεγάλος και ειδικά αν το πρόβλημά μας αφορά μη-κυρτή βελτιστοποίηση τότε η εφαρμογή του αλγορίθμου gradient descent είναι απολύτως μη-πρακτική. Φαίνεται λοιπόν ότι χρειαζόμαστε μία τροποποίηση προκειμένου να είμαστε σε θέση να λύσουμε αποδοτικά ένα πρόβλημα μηχανικής μάθησης.

4.10 Στοχαστικός Αλγόριθμος Gradient Descent και Mini-Batch Gradient Descent

Ο αλγόριθμος του στοχαστικού gradient descent είναι μία επέκταση του κανονικού αλγορίθμου που περιγράφηκε στην ενότητα **4.8**. Η ανάγκη της ανάπτυξης του εν λόγω αλγορίθμου έγκειται στις δυσκολίες που παρουσιάζονται στην εφαρμογή του απλού gradient descent όταν ο αριθμός των δεδομένων εκπαίδευσης είναι πάρα πολύ μεγάλος. Πιο συγκεκριμένα, όπως είδαμε στην ενότητα **4.6**, οι συναρτήσεις κόστους αποτελούν αθροίσματα (τετραγωνικών ή και άλλου είδους) σφαλμάτων μεταξύ των δεδομένων εξόδου και των προβλέψεών μας. Όταν έχουμε στην διάθεσή μας έναν υπερβολικά μεγάλο αριθμό δεδομένων (της τάξεως των εκατομμυρίων) τότε προφανώς ένα μονάχα βήμα του αλγορίθμου gradient descent απαιτεί πάρα πολύ μεγάλο υπολογιστικό χρόνο.

Η καρδιά του στοχαστικού gradient descent είναι η εφαρμογή του κλασσικού gradient descent με την παραλλαγή ότι σε κάθε επανάληψη θα υπολογίζουμε την βαθμίδα *για ένα μόνο δεδομένο*. Με άλλα λόγια, στην i-οστή επανάληψη του αλγορίθμου, αντί να υπολογίσουμε την βαθμίδα $\sum_{i=1}^{n} \nabla_\theta \mathcal{L}_i$ θα πρέπει μόνο να υπολογίσουμε την βαθμίδα $\nabla_\theta \mathcal{L}_i$. Με την μέθοδο αυτή ο υπολογιστικός χρόνος μειώνεται σημάντικα αφού σε κάθε επανάληψη θα πρέπει να υπολογίσουμε πολύ πιο απλές βαθμίδες σε σχέση με την κλασσική έκδοση του αλγορίθμου.

Μία μικρή παραλλαγή του στοχαστικού gradient descent είναι ο λεγόμενος αλγόριθμος *mini-batch gradient descent*. Στην περίπτωση αυτή κάνουμε μία τυχαία επιλογή *παρτίδων* (mini-batches) σταθερού μεγέθους $|\mathcal{B}|$ από τα δεδομένα μας και εφαρμόζουμε τον αλγόριθμο gradient descent σε αυτά. Σε κάθε επανάληψη δηλαδή θα πρέπει να υπολογισθεί η βαθμίδα

$$\nabla_{\boldsymbol{\vartheta}} \mathbf{J}(\boldsymbol{\vartheta}) \ = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \nabla_{\boldsymbol{\vartheta}} \mathcal{L}(\boldsymbol{\vartheta})_i$$

Ο υπολογιστικός χρόνος σε αυτήν την περίπτωση είναι $\mathcal{O}(\mathcal{B})$.

4.11 Προς-τα-πίσω Διάδοση (Back Propagation)

Έχοντας τώρα τον αλγόριθμο που θα χρησιμοποιήσουμε (gradient descent) για την ελαχιστοποίηση του κόστους θα πρέπει να σκεφτούμε πώς θα υπολογίσουμε τις μερικές παραγώγους (βαθμίδες) που απαιτεί. Η μεθοδολογία υπολογισμού αυτών των μερικών παραγώγων ονομάζεται *προς-τα-πίσω διάδοση ή back-propagation* και επιτρέπει την ροή πληροφορίας που παίρνουμε από τη συνάρτηση κόστους *προς τα πίσω* κατά μήκος του νευρωνικού δικτύου. Η ροή αυτή μοντελοποιείται με βάση τον γνώστο από τον Μαθηματικό Λογισμό *κανόνα της αλυσίδας*. Έστω $\mathbf{x} \in \mathbb{R}^m$, $\mathbf{y} \in \mathbb{R}^n$, $g : \mathbb{R}^m \to \mathbb{R}^n$ και $f : \mathbb{R}^n \to \mathbb{R}$. Απαιτούμε $g \in C^1(\mathbb{R}^m)$ και $f \in C^1(\mathbb{R})^n$. Αν $\mathbf{y} = g(\mathbf{x})$ και $z = f(\mathbf{y})$ τότε:

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \mathcal{J}_{ij}$$

όπου $\mathcal{J}_{ij}$ ο *Ιακωβιανός πίνακας* της g. Σε διανυσματική μορφή ο κανόνας της αλυσίδας γράφεται ως

$$\nabla_{\mathbf{x}} z = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}^{\intercal}\right) \nabla_{\mathbf{y}} z$$

Δηλαδή η μερική παράγωγος ως προς μία μεταβλητή $\mathbf{x}$ μπορεί να υπολογισθεί πολλαπλασιάζοντας τον Ιακωβιανό πίνακα $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ με την βαθμίδα $\nabla_{\mathbf{y}} z$. Ο αλγόριθμος back-propagaton δεν εφαρμόζεται καθαρά και μόνο σε διανύσματα αλλά και γενικά σε τανυστές οποιασδήποτε διάστασης. Πριν συνεχίζουμε με την περιγραφή του αλγορίθμου back-propagation στα πλαίσια των συμβολισμών ενός νευρωνικού δικτύου θα πρέπει να εισάγουμε έναν νέο συμβολισμό, το *γινόμενο Hadamar*.

Ορισμός: Για δύο πίνακες $\mathbf{A}$ και $\mathbf{B}$ ίδιας διάστασης $m \times n$, το *γινόμενο Hadamar* $\mathbf{A} \odot \mathbf{B}$ είναι ένας πίνακας ίδιας διάστασης με στοιχεία που δίνονται από την σχέση

$$(\mathbf{A} \odot \mathbf{B})_{ij} = (\mathbf{A})_{ij} (\mathbf{B})_{ij}$$

Πρόκειται δηλαδή για πολλαπλασιασμό ανά στοιχείο. Για παράδειγμα για δύο 2×2 πίνακες το γίνομενο Hadamar ορίζεται ως

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \odot \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{bmatrix} \tag{4.11.1}$$

Το γινόμενο Hadamar είναι αντιμεταθετικό, προσεταιριστικό και επιμεριστικό με πράξη την πρόσθεση πινάκων.

Ο αλγόριθμος back-propagation μας δείχνει πώς αλλάζοντας τα βάρη και τα biases ενός νευρωνικού δικτύου αλλάζει η συνάρτηση κόστους. Αυτό εν τέλει σημαίνει προφανώς πως θα πρέπει να υπολογίσουμε τις μερικές παραγώγους $\frac{\partial C}{\partial w^l_{jk}}$ και $\frac{\partial C}{\partial b^l_j}$. Για να υπολογίσουμε αυτές τις παραγώγους θα πρέπει να κάνουμε άλλον έναν ενδιάμεσο υπολογισμό. Ο υπολογισμός αυτός περιλαμβάνει το λεγόμενο *σφάλμα* (error), δ^l_j. Η ποσότητα αυτή αποτελεί το σφάλμα στον j-οστό νευρώνα του l-οστού στρώματος. Ο αλγόριθμος back-propagation μας δίνει ακριβώς έναν τρόπο για να υπολογίσουμε το σφάλμα δ^l_j, το οποίο σχετίζεται άμεσα με τις μερικές παραγώγους $\frac{\partial C}{\partial w^l_{jk}}$ και $\frac{\partial C}{\partial b^l_j}$.

Ας υποθέσουμε ότι z^l_j είναι η σταθμησμένη είσοδος (weighted input) του j-οστού νευρώνα του l-οστού στρώματος. Τότε το σφάλμα ορίζεται ως

$$\delta^l_j \equiv \frac{\partial C}{\partial z^l_j}$$

Με βάση τις γνωστές συμβάσεις θα συμβολίσουμε με δ^l το διάνυσμα των σφαλμάτων που σχετίζεται με το στρώμα l. Η εξίσωση που δίνει το σφάλμα για το στρώμα εξόδου είναι η εξής:

$$\delta^L_j = \frac{\partial C}{\partial \alpha^L_j} \sigma'(z^L_j) \tag{4.11.2}$$

Ο πρώτος όρος στο δεξί μέλος, $\frac{\partial C}{\partial \alpha_j^L}$, μετράει πόσο γρήγορα αλλάζει η συνάρτηση κόστους ως συνάρτηση της j-οστής ενεργοποίησης εξόδου. Ο δεύτερος όρος , $\sigma'(z_j^L)$ μετράει πόσο γρήγορα η συνάρτηση ενεργοποίησης σαλλάζει στο z_j^L. Η έκφραση αυτή δεν είναι παρά ο κάνονας της αλυσίδας. Πράγματι

$$\delta_j^L = \frac{\partial C}{\partial z_j^L} = \frac{\partial C}{\partial \alpha_j^L}\frac{\partial \alpha_j^L}{\partial z_j^L}$$

Όμως $\alpha_j^L = \sigma(z_j^L)$ και άρα $\frac{\partial \alpha_j^L}{\partial z_j^L} = \sigma'(z_j^L)$. Έτσι παίρνουμε την παραπάνω έκφραση. Η εξίσωση (5.0.2) μπορεί να γραφτεί σε μία πιο συμπυκνωμένη μορφή πινάκων ως εξής

$$\delta^L = \nabla_\alpha C \odot \sigma'(z^L) \tag{4.11.3}$$

όπου $\nabla_\alpha C$ είναι το διάνυσμα του οποίου οι συνιστώσες είναι οι μερικές παράγωγοι $\frac{\partial C}{\partial \alpha_j^L}$. Εκφράζει τον ρυθμό μεταβολής της συνάρτησης C ως προς τις τιμές ενεργοποίησης εξόδου. Παρακάτω δίνουμε την σχέση που εκφράζει το σφάλμα δ^l συναρτήσει του σφάλματος του επόμενου στρώματος δ^{l+1}:

$$\delta^l = \left[(w^{l+1})^\intercal \delta^{l+1}\right] \odot \sigma'(z^l) \tag{4.11.4}$$

όπου $(w^{l+1})^\intercal$ είναι ο ανάστροφος πίνακας του πίνακα βαρών w^{l+1} του $(l+1)$-οστού στρώματος. Μία ίσως διαισθητική εξήγηση της παραπάνω εξίσωσης είναι η εξής [9]: Ας υποθέσουμε ότι γνωρίζουμε το σφάλμα δ^{l+1}. Τότε πολλαπλασιάζοντας από τα αριστερά με τον πίνακα w^{l+1} μεταφέρουμε το σφάλμα προς τα πίσω κατά μήκος του δικτύου, παίρνοντας έτσι μία εκτίμηση του σφάλματος της εξόδου του l-οστού στρώματος. Έπειτα παίρνουμε το γινόμενο Hadamar $\odot\sigma'(z^l)$. Η πράξη αυτή μεταφέρει το σφάλμα κι άλλο προς τα πίσω δια μέσου της συνάρτησης ενεργοποίησης του l-οστού στρώματος, δίνοντάς μας έτσι το σφάλμα δ^l της σταθμησμένης εισόδου του l-οστού στρώματος.

Με την βοήθεια των εξισώσεων (4.11.2) και (4.11.4) μπορούμε να υπολογίσουμε το σφάλμα δ^l για κάθε στρώμα του δικτύου ξεκινώντας από το στρώμα εξόδου του δικτύου και ταξιδεύοντας προς τα πίσω, προς το στρώμα εισόδου του δικτύου. Εξ'ου και το όνομα *προς-τα-πίσω* διάδοση. Στο σημείο αυτό είμαστε έτοιμοι τώρα να δούμε πως θα υπολογίσουμε τις μερικές παραγώγους-στόχους, δηλαδή τις μερικές παραγώγους ως προς τις παραμέτρους του δικτύου, δηλαδή τα βάρη και τα biases. Ξεκινάμε δίνοντας την εκπληκτικά απλή εξίσωση που δίνει την μερική παράγωγο της συνάρτησης κόστους ως προς τα βάρη του δικτύου

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{4.11.5}$$

Η μερική παράγωγος ως προς τα βάρη είναι, προς μεγάλη μας έκπληξη και χαρά το σφάλμα, για το οποίο έχουμε ήδη βρει τρόπο υπολογισμού. Αν δεν υπάρχει σύγχηση ως προς τους δείκτες μπορούμε να γράψουμε την παραπάνω εξίσωση σε μία πιο συμπυκνωμένη μορφή

$$\frac{\partial C}{\partial b} = \delta \tag{4.11.6}$$

όπου φυσικά το σφάλμα δυπολογίζεται στον ίδιο νευρώνα με το bias b. Συνεχίζουμε με την εξίσωση που δίνει την μερική παράγωγο της συνάρτηση κόστους ως προς ένα βάρος του δικτύου

$$\frac{\partial C}{\partial w^l_{jk}} = \alpha^{l-1}_k \delta^l_j \tag{4.11.7}$$

Φυσικά γνωρίζοντας ήδη πως να υπολογίσουμε τις ποσότητες α^{l-1}_k και δ^l_j μπορούμε εύκολα να υπολογίσουμε και την μερική παράγωγο $\frac{\partial C}{\partial w^l_{jk}}$. Και πάλι, μπορούμε να γράψουμε την παραπάνω εξίσωση στην πιο συμπυκνωμένη και πιο διαισθητική μορφή

$$\frac{\partial C}{\partial w} = \alpha_{in}\delta_{out} \tag{4.11.8}$$

όπου φυσικά η α_{in} είναι η ενεργοποίηση του νευρώνα που έχει σαν είσοδο το βάρος w και δ_{out} είναι το σφάλμα του νευρώνα που έχει σαν έξοδο το βάρος w. Παρατηρούμε ότι όταν η τιμή α_{in} είναι πολύ μικρή, δηλαδή $\alpha_{in} \approx 0$, η μερική παράγωγος $\frac{\partial C}{\partial w}$ θα είναι εξίσου μικρή. Τότε λέμε ότι αυτό το βάρος *μαθαίνει αργά*, που σημαίνει ότι δεν αλλάζει πολύ κατά την διάρκεια του gradient descent. Στον παρακάτω πίνακα παρουσιάζονται συγκεντρωμένες οι εξισώσεις που συνιστούν την διαδικασία του back-propagation [9]:

Οι εξισώσεις του αλγορίθμου Back-Propagation,

$$\delta^L = \nabla_\alpha C \odot \sigma'(z^L) \tag{1}$$

$$\delta^l = \left[(w^{l+1})^\intercal \delta^{l+1}\right] \odot \sigma'(z^l) \tag{2}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \tag{3}$$

$$\frac{\partial C}{\partial w^l_{jk}} = \alpha^{l-1}_k \delta^l_j \tag{4}$$

Οι εξισώσεις αυτές φυσικά αποδεικνύονται με αυστηρό μαθηματικό τρόπο. Οι αποδείξεις είναι απλές και αποτελούν απλή και προσεχτική εφαρμογή του κανόνα της αλυσίδας. Οι αποδείξεις παρατίθενται στο παράρτημα.

Έχοντας τώρα περιγράψει τόσο την διαδικασία της προώθησης όσο και της προς-τα-πίσω διάδοσης, δηλαδή μία πλήρη εποχή της εκπαίδευσης ενός νευρωνικού δικτύου, μπορούμε εύκολα να την συνοψίσουμε σέ έναν μικρό αλγόριθμο. Στον παρακάτω πίνακα δίνεται συνοπτικά ο συνολικός αλγόριθμος της προώθησης και της προς-τα-πίσω διάδοσης [9].

Αλγόριθμος Προώθησης και Προς-τα-πίσω Διάδοσης
1. **Διάνυσμα Εισόδου x**: Υπολογίζουμε το αντίστοιχο διάνυσμα ενεργοποιήσεων α^1 του στρώματος εισόδου 2. **Προώθηση**: Για κάθε $n = 1, 2, \cdots, L$ υπολογίζουμε τις ποσότητες $z^n = w^n \alpha^{n-1} + b^n$ και $\alpha^n = \sigma(z^n)$ 3. **Σφάλμα Εξόδου** δ^L: Υπολογίζουμε το διάνυσμα $\delta^L = \nabla_\alpha C \odot \sigma'(z^L)$ 4. **Προς-τα-πίσω διάδοση του σφάλματος**: Για κάθε $l = L-1, L-2, \cdots, 1$ υπολογίζουμε το σφάλμα $\delta^l = \left[(w^{l+1})^\intercal \delta^{l+1}\right] \odot \sigma'(z^l)$ 5. **Έξοδος**: Υπολογίζουμε τις μερικές παραγώγους της συνάρτησης κόστους από τις σχέσεις $\frac{\partial C}{\partial b^l_j} = \delta^l_j$ και $\frac{\partial C}{\partial w^l_{jk}} = \alpha^{l-1}_k \delta^l_j$

Πίνακας 2: Αλγόριθμος Feed-Forward και Back-Propagation

Θα λήξουμε το κεφάλαιο με ένα πολύ σημαντικό θεώρημα της επιστήμης των νευρωνικών δικτύων που μας δείχνει και την ευρωστία και τις ικανότητες εν γένει της αρχιτεκτονικής των νευρωνικών δικτύων να δρουν ως συναρτησιακοί προσεγγιστές.

4.12 Το Θεώρημα Καθολικής Προσέγγισης (Universal Approximation Theorem)

Ένας τεράστιος αριθμός σύγχρονων τεχνολογικών και επιστημονικών εφαρμογών βασίζονται στην ικανότητα των νευρωνικών δικτύων να συσχετίζουν σχεδόν κάθε είδους δεδομένων εισόδου με μία επιθυμητή έξοδο, δηλαδή στο να βρίσκουν μία συνάρτηση που να τα συσχετίζει. Πιο σωστά, τα νευρωνικά δίκτυα δρούν ως *συναρτησιακοί προσεγγιστές* (function approximators), με την έννοια ότι μπορούν να "πλησιάσουν" μια επιθυμητή συνάρτηση με πολύ μεγάλη ακρίβεια. Πώς όμως μπορεί κανείς να είναι σίγουρος ότι η αρχιτεκτονική των νευρωνικών δικτύων θα είναι όντως ικανή να επιλύσει το εκάστοτε πρόβλημα; Στα πλαίσια των πραγματικών εφαρμογών οι συναρτήσεις που επιθυμούμε να προσεγγίσουμε μπορεί να είναι πολύ περίπλοκες ή ακόμη και να μην έχουν σαφή (explicit) μορφή.

Στο ερώτημα αυτό δίνει μία πολύ ικανοποιητική απάντηση το πανίσχυρο *Θεώρημα Καθολικής Προσέγγισης για Νευρωνικά Δίκτυα* (Universal Approximation Theorem for Neural Networks) [14]. Το θεώρημα αυτό έχει μία πολύ καθορισμένη και αυστηρή Μαθηματική θεμελίωση. Μας λέει πως μπορούμε να προσεγγίσουμε όσο καλά θέλουμε μια συνεχή συνάρτηση ορισμένη σε ένα συμπαγές σύνολο με ένα προωθητικό νευρωνικό δίκτυο που περιέχει *μονάχα ένα κρυφό στρώμα* το οποίο θα αποτελείται από *πεπερασμένους νευρώνες*. Η απόδειξη του θεωρήματος δεν είναι απλή και χρησιμοποιεί πολλά βαριά εργαλεία της συναρτησιακής ανάλυσης και της θεωρίας μέτρου, όπως το θεώρημα Hahn-Banach και το Θεώρημα Κυρίαρχης Σύγκλισης. Παρόλα αυτά μας εξασφαλίζει την αδιαμφησβήτητη ικανότητα των νευρωνικών δικτύων να μας παρέχουν, με μία σχετικά απλή αρχιτεκτονική, οποιαδήποτε ομοιόμορφα συνεχή συνάρτηση θέλουμε. Πρίν δοθεί η πλήρης Μαθηματική περιγραφή του θεωρήματος θα χρειαστούμε μερικούς ορισμούς.

Ορισμός: Μία συνάρτηση σ: $\mathbb{R} \to \mathbb{R}$ θα λέγεται *σιγμοειδής* (sigmoidal) αν είναι γνήσια μονότονη και ικανοποιεί:

$$\sigma(t) = \begin{cases} \alpha & ,t \to +\infty \\ \beta & ,t \to -\infty \end{cases}$$

για κάποιους πραγματικούς αριθμούς α και β. Δηλαδή μία σιγμοειδής συνάρτηση είναι φραγμένη. Στις περισσότερες εφαρμογές και πολύ συνχά στην βιβλιογραφία οι σιγμοειδείς συναρτήσεις επιλέγονται με τέτοιο τρόπο ώστε να είναι κανονικοποιημένες, δηλαδή α = 1 και β = 0 ή α = 1 και β = -1.

Παραδείγματα σιγμοειδών συναρτήσεων αποτελούν οι εξής συναρτήσεις :

- *Λογιστική Συνάρτηση*, σ(x) = $\frac{1}{1+e^{-x}}$
- *Υπερβολική Εφαπτομένη* σ(x) = $\frac{e^x - e^{-x}}{e^x + e^{-x}}$
- *Συνάρτηση Σφάλματος* σ(x) = $\frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$

Ορισμός:Έστω I_n ο μοναδιαίος n-διάστατος κύβος. Μία συνάρτηση σ θα λέγεται *μεροληπτική* (discriminatory) εάν ισχύει η συναπαγωγή:

$$\int_{I_n} \sigma(\mathbf{w}_i^\intercal \mathbf{x} + b_i) d\mu(x) = 0 \Rightarrow \mu(x) = 0 \tag{4.12.1}$$

$\forall$ $\mathbf{w}_i \in \mathbb{R}^d$, $b_i \in \mathbb{R}$, $\mu \in M(I_n)$.

Η συνάρτηση μ(x) αποτελεί ένα κάπως τεχνικό εργαλείο και καλείται *μέτρο*. Είναι μια συνάρτηση που ορίζεται σε ένα σύνολο και μας επιτρέπει να αντιστοιχίσουμε σε κάθε υποσύνολο του συνόλου αυτού έναν θετικό πραγματικό αριθμό, ο οποίος διαισθητικά μπορεί να ερμηνευθεί ώς το *μέγεθος* του υποσυνόλου αυτού. Το σύνολο $M(I_n)$ μπορούμε να το σκεφτούμε ως ένα σύνολο στο οποίο έχει νοήμα να ορίσουμε συναρτήσεις-μέτρα. Είναι το σύνολο των μέτρων ορισμένα στον n-διάστατο μοναδιαίο κύβο. Σε κάθε πιθανό υποσύνολο του κύβου αυτού αντιστοιχούμε ένα μέτρο, το οποίο μας δίνει τον n-διάστατο όγκο του. Ο παραπάνω ορισμός μας λέει πως μία μεροληπτική συνάρτηση μπορεί να δράσει *μη-καταστρεπτικά* σε έναν γραμμικό συνδυασμό πραγματικών αριθμών με εξαίρεση ένα υποσύνολο που έχει μέτρο ίσο με το μηδέν (μ= 0). Διαισθητικά, ένα σύνολο έχει μέτρο ίσο με το μηδέν όταν ο πληθάριθμός του είναι πεπερασμένος. Συνεπώς μια μεροληπτική συνάρτηση επιτρέπει στην πληροφορία να περάσει από τον έναν νευρώνα στον επόμενο χωρίς να χαθεί η πληροφορία της εισόδου.

Έστω A $\subset \mathbb{R}$ ένα συμπαγές σύνολο (κλειστό και φραγμένο). Θα συμβολίζουμε το σύνολο των συνεχών, μεροληπτικών και μη γραμμικών σιγμοειδών συναρτήσεων ορισμένες

στο A με παραμέτρους α και β ώς **Sig(A,α,β)**. Επειδή το A είναι συμπαγές και οι συναρτήσεις συνεχείς σημαίνει ότι το Sig(A,α,β) αποτελείται από ομοιόμορφα συνεχείς -και άρα ολοκληρώσιμες- συναρτήσεις.

Παρακάτω δίνουμε έναν μαθηματικό ορισμό ενός νευρωνικού δικτύου με ένα μόνο κρυφό στρώμα:

Ορισμός: Ένα *νευρωνικό δίκτυο* N νευρώνων (ή κόμβων) διατεταγμένων σε ένα μόνο κρυφό στρώμα είναι μία συνάρτηση $\Psi : \mathbb{R}^n \to \mathbb{R}$ που δίνεται από την σχέση:

$$\Psi(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i \sigma(\mathbf{w}_i^\intercal \mathbf{x} + b_i) \tag{4.12.2}$$

όπου $\mathbf{w}_i, \mathbf{x} \in \mathbb{R}^n, \lambda_i, b_i \in \mathbb{R}$ και σ∈ Sig(I_n,α,β). Το σύνολο των νευρωνικών δικτύων της παραπάνω μορφής που αποτελούνται από ένα κρυφό στρώμα θα συμβολίζεται με $\aleph_1$. Ο όρος που υπάρχει μέσα στο άθροισμα του παραπάνω ορισμού μας θυμίζει φυσικά την *έξοδο ενός νευρώνα*, όπως παρουσιάσθηκε στην ενότητα **4.4**. Η παραπάνω συνάρτηση Ψ λοιπόν δεν είναι παρά ένας γραμμικός συνδυασμός των σημάτων εξόδου όλων των νευρώνων του μοναδικού κρυφού στρώματος του δικτύου. Το άθροισμα αυτό αποτελεί φυσικά την έξοδο του δικτύου.

Θεώρημα (Καθολικής Προσέγγισης για Νευρωνικά Δίκτυα) [14]: Έστω $\sigma : \mathbb{R} \to \mathbb{R}$ με σ∈ Sig(I_n,α,β) και C(I_n) το σύνολο των συνεχών συναρτήσεων στο I_n (ή σε οποιοδήποτε συμπαγές υποσύνολο του $\mathbb{R}^n$. Τότε το σύνολο $\aleph_1$ είναι *πυκνό* στο C(I_n). Δηλαδή για κάθε $\epsilon > 0$ και κάθε συνάρτηση $f \in C(I_n)$ υπάρχει ένα νευρωνικό δίκτυο της μορφής (4.12.2) τέτοιο ώστε:

$$|\Psi(x) - f(x)| < \epsilon$$

$\forall x \in I_n$.

Από το παραπάνω κεντρικό θεώρημα φαίνεται ότι η αρχιτεκτονική των νευρωνικών δικτύων είναι παραπάνω από ικανή για να δράσει ώς συναρτησιακός προσεγγιστής. Ένα δίκτυο με ένα μόνο κρυφό στρώμα αρκεί για να μας εξασφαλίσει όλες τις συνεχείς και φραγμένες συναρτήσεις, πόσο μάλλον ένα "βαθύτερο" δίκτυο με παραπάνω κρυφά στρώματα.

Στο σημείο αυτό έχουμε ολοκληρώσει το μεγάλο αυτό κεφάλαιο του Deep Learning. Όπως είναι προφανές η εκπαίδευση ενός τεχνητού νευρωνικού δικτύου δεν είναι εύκολη υπόθεση και αποτελεί ένα πρόβλημα μη-κυρτής βελτιστοποίησης που όμως βρίσκει πάρα πολλές εφαρμογές στην σύγχρονη τεχνολογία. Ακολουθεί ένα κεφάλαιο εξίσου σημαντικό, αυτό της *ενισχυτικής μάθησης*. Είναι μία από τις δύο θεωρίες (μαζί με το Deep Learning) που όταν συνδυαστούν μας δίνουν ένα πολύ χρήσιμο εργαλείο της τεχνητής νοημοσύνης, την λεγόμενη *ενισχυτική μάθηση βάθους*.

5 Ενισχυτική Μάθηση

5.1 Διαδικασία Απόφασης κατά Markov

Η *Ενισχυτική Μάθηση* (Reinforcement Learning) είναι ένας κλάδος της μηχανικής μάθησης που λειτουργεί μέσω ακολουθιακής λήψης αποφάσεων. Ο μαθηματικός φορμαλισμός της ενισχυτικής μάθησης περιλαμβάνει τη λεγόμενη *Διαδικασία Απόφασης κατά Markov* (Markov Decision Process). Η μεθοδολογία αυτή μελετά την αλληλεπίδραση ενός *πράκτορα* (agent) με το *περιβάλλον* (enviroment) στο οποίο ανήκει. Σκοπός του πράκτορα είναι εν τέλει να μεταβεί σε μία επιθυμητή τελική *κατάσταση* (state). Πιο συγκεκριμένα, ο πράκτορας είναι σε θέση να επιλέξει μία από τις επιτρεπόμενες *δράσεις* (actions) έτσι ώστε να μεταβεί από την τρέχουσα κατάσταση στην οποία βρίσκεται στην επόμενη. Αφού γίνει αυτή η μετάβαση, ο πράκτορας αξιολογείται μέσω μίας *αμοιβής* (reward). Η αμοιβή αυτή στα πλαίσια της μηχανικής μάθηση είναι αυθαίρετη. Για παράδειγμα, κάθε φορά που ο πράκτορας μεταβαίνει σε μία κατάσταση από την οποία δεν επωφελείται ή απομακρύνεται από την τελική επιθυμητή κατάσταση, τότε δίνουμε αμοιβή 0 ή -1 ενώ κάθε φορά που ο πράκτορας μεταβαίνει σε μία κατάσταση που τον φέρνει πιο κόντα στον τελικό στόχο τότε δίνουμε αμοιβή +1. Για μερική ευκολία στην κατανόηση, μπορούμε να κάνουμε τον εξής παραλληλισμό:

Ας φανταστούμε ότι ο πράκτορας είναι, για παράδειγμα, ένα κατοικίδιο το οποίο θέλουμε να σωφρονίσουμε. Κάθε φορά που το κατοικίδιο δεν ησυχάζει όταν του το ζητάμε τότε δεν του δίνουμε σημασία (αμοιβή 0) ή το βγάζουμε έξω από το σπίτι (αμοιβή -1). Σε αντίθετη περίπτωση, όταν το κατοικίδιο αρχίσει να καταλαβαίνει την εντολή μας και ησυχάζει όταν του το ζητάμε τότε το επιβραβεύουμε με κάτι που του αρέσει (π.χ. ένα φαγητό που του αρέσει, αμοιβή +1). Το παράδειγμα αυτό είναι εξαιρετικά απλοϊκό αφού εδώ οι πιθανές καταστάσεις είναι μόνο δύο (το κατοικίδιο είναι είτε ήσυχο είτε όχι) και οι πιθανές δράσεις είναι επίσης δύο (είτε να υπακούσει στην εντολή μας είτε όχι). Η βασική ιδέα όμως είναι αυτή.

Έχοντας τα παραπάνω, είμαστε έτοιμοι τώρα να διατυπώσουμε πιο φορμαλιστικά μία Διαδικασία Απόφασης κατά Markov (στο εξής MDP) [4]. Αποτελεί μία διακριτή στοχαστική διαδικασία ελέγχου και διατυπώθηκε για πρώτη φορά από τον Richard Ernest Bellman. Η ιδέα αυτή, παρόλο που προς το παρόν θα ορισθεί σε διακριτά σύνολα, μπορεί να επεκταθεί και σε συνεχείς χώρους.

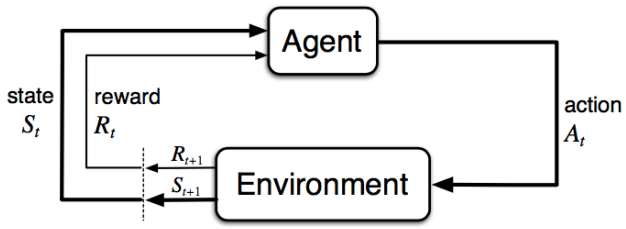
Ορισμός: Μία (διακριτή) *Διαδικασία Απόφασης κατά Markov* (MDP) είναι μία πεντάδα $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$ όπου:

- $\mathcal{S}$ ο *Χώρος Καταστάσεων* (State Space)
- $\mathcal{A}$ ο *Χώρος Δράσεων* (Action Space)
- $\mathcal{P} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ η φραγμένη *συνάρτηση μετάβασης* (transition function) που περιλαμβάνει το σύνολο των δεσμευμένων πιθανοτήτων μετάβασης μεταξύ των καταστάσεων.
- $R : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ η *συνάρτηση κέρδους ή αμοιβής* (reward function) και $\mathcal{R} \subset \mathbb{R}^+$

- $\gamma \in [0, 1)$ ο *παράγοντας προεξόφλησης* (discount factor).

Σε κάθε χρονικό βήμα t η πιθανότητα μετάβασης από την κατάσταση s_t στην επόμενη κατάσταση, s_{t+1}, μέσω την λήψης της απόφασης α_t δίνεται από την συνάρτηση $\mathcal{P}(s_t, s_{t+1}, \alpha_t)$, ενώ η αντίστοιχη αμοιβή από την συνάρτηση $\mathrm{R}(s_t, s_{t+1}, \alpha_t)$. Όσον αφορά την συνάρτηση μετάβασης, θεωρούμε ότι μία MDP ικανοποιεί την *ιδιότητα Markov*. Στην ουσία, η πιθανότητα να μεταβούμε σε μια επόμενη κατάσταση, s_{t+1}, εξαρτάται μόνο από το άμεσο παρελθόν της διαδικασίας, δηλαδή μόνο από την κατάσταση του προηγούμενου χρονικού βήματος, s_t, δηλαδή $\mathcal{P}(s_{t+1}|s_1, \alpha_1, \cdots, s_t, \alpha_t) = \mathcal{P}(s_{t+1}|s_t, \alpha_t)$. Ο ρόλος του παράγοντα προεξόφλησης γ θα αναλυθεί παρακάτω.

Διαγραμματικά, μία MDP φαίνεται στο παρακάτω σχήμα:



Εικόνα 6: Μία τυπική Διαδικασία Απόφασης κατά Markov

Όπως είναι προφανές, σκοπός μίας MDP είναι να μεταβούμε σε μία επιθυμητή τελική κατάσταση αρχίζοντας από μία δεδομένη, γνωστή κατάσταση. Εφόσον οι μεταβάσεις μεταξύ των καταστάσεων γίνεται μέσω των διαθέσιμων δράσεων σημαίνει ότι, ιδανικά, θα πρέπει να προσδιορίσουμε μία συνάρτηση:

$$\mu : \mathcal{S} \to \mathcal{A}$$

Η συνάρτηση μ ονομάζεται *πολιτική ή στρατηγική* (policy) και θα είναι στην ουσία ο οδηγός του πράκτορα δια μέσου της MDP, δηλαδή θα είναι ένας κανόνας επιλογής δράσης σε μία δεδομένη κατάσταση. Μπορούμε ίσως να κατανοήσουμε καλύτερα την έννοια της συνάρτησης πολιτικής κάνοντας έναν μικρό παραλληλισμό με το σκάκι. Στις περισσότερες περιπτώσεις τα άτομα που μαθαίνουν σκάκι σε μεγάλο βαθμό και εις βάθος, έρχονται σε επαφή με μεγάλη ποικιλία εκπαιδευτικού υλικού, στο οποίο περιέχονται οι λεγόμενες *βαριάντες*. Μία βαριάντα είναι ένα σύνολο από κινήσεις οι οποίες έχουν αποδειχθεί πολύ χρήσιμες σε κάποιο από τα στάδια του παιχνιδιού. Για παράδειγμα όσον αφορά την αρχή ενός παιχνιδιού, το λεγόμενο *άνοιγμα*, πολύ γνωστές είναι οι βαριάντες "Γαλλική Άμυνα" και η "Βαριάντα Τσιγκορίν". Είναι σύνολα από συγκεκριμένες κινήσεις (ή δράσεις) που οδηγούν σε μία στρατηγικά συμφέρουσα κατάσταση ανοίγματος. Αυτός ακριβώς είναι και ο ρόλος της συνάρτησης πολιτικής ή στρατηγικής στα πλαίσια της ενισχυτικής μάθησης, να παρέχει στον πράκτορα ένα σύνολο από δράσεις, οι οποίες θα τον οδηγήσουν επιτυχώς προς την επιθυμητή τελική κατάσταση (όπως οι βαριάντες οδηγούν τον παίχτη σκακιού στην νίκη). Θα εξετάσουμε αργότερα με ποια κριτήρια θα πρέπει να προσδιορισθεί η συνάρτηση αυτή.

Επίσης κρίνεται σκόπιμο να αναφερθεί ότι οι συναρτήσεις πολιτικής διακρίνονται σε *ντετερμινιστικές*, όπως η παραπάνω, και σε *μη-ντετερμινιστικές* [4]. Στην δεύτερη περίπτωση οι συναρτήσεις πολιτικής ορίζονται ως:

$$\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$$

Συνήθως στην βιβλιογραφία οι ντετερμινιστικές πολιτικές συμβολίζονται με μ ενώ οι μη-ντετερμινιστικές με π. Αυτόν τον συμβολισμό θα ακολουθήσουμε και εδώ. Οι ντετερμινιστικές συναρτήσεις πολιτικής καθορίζουν πλήρως ποια δράση είναι η καταλληλότερη για την δεδομένη κατάσταση, ενώ οι μη-ντετερμινιστικές συναρτήσεις πολιτικής έχουν στοχαστική λειτουργία και μας δίνουν την πιθανότητα να επιλεγεί μία δράση α όταν ο πράκτορας βρίσκεται σε μία κατάσταση s.

Τώρα που έχουμε έναν στόχο, δηλαδή τον προσδιορισμό της συνάρτησης π (ή μ), θα πρέπει να σκεφτούμε με τι κριτήρια θα την επιλέξουμε. Για τον σκοπό αυτό χρειαζόμαστε τους εξής ορισμούς [15]:

Ορισμός : Το *Αναμενόμενο Κέρδος* (Expected Return) μίας MDP ορίζεται ως:

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

όπου $R_t = R(s_t, \alpha, s_{t+1})$. Είναι δηλαδή η τιμή του *προεξοφλημένου* κέρδους (discounted reward) που λαμβάνει ο πράκτορας ξεκινώντας από μία κατάσταση s_t και ακολουθώντας μία συγκεκριμένη δράση.

Ο παράγοντας γ παίζει διπλό ρόλο. Καταρχάς, το γεγονός ότι είναι μία σταθερή τιμή μικρότερη της μονάδας επιτρέπει στο παραπάνω άπειρο άθροισμα να συγκλίνει ευκολότερα, εφόσον στις περισσότερες περιπτώσεις η συνάρτηση αμοιβής είναι και αυτή μια φραγμένη συνάρτηση. Η φυσική του σημασία είναι η εξής: Καθώς $\gamma \to 0$ (και κάνοντας τη σύμβαση ότι $0^0 = 1$) παρατηρούμε ότι όλοι οι όροι του αθροίσματος πέραν του R_{t+1} μηδενίζονται. Αυτό σημαίνει ότι ο πράκτορας δρα "μυωπικά", δηλαδή σκοπός του είναι να βρει τρόπο να πάρει το μέγιστο δυνατό κέρδος στο επόμενό του βήμα, χωρίς να έχει μακροπρόθεσμες βλέψεις. Εν αντιθέσει, καθώς $\gamma \to 1$ ο πράκτορας δίνει όλο και περισσότερο βάρος σε μελλοντικές αμοιβές και δρά έτσι ώστε να έχει μακρυπρόθεσμα το μέγιστο κέρδος.

Τέλος δεν είναι δύσκολο να δείξει κανείς ότι:

$$G_t = R_{t+1} + \gamma G_{t+1}$$

Ορισμός: (*Συναρτήσεις Αναμενόμενου Κέρδους* V^π *και* Q^π). Η συνάρτηση V^π (V-value function) ορίζεται ως εξής:

$$V^\pi : \mathcal{S} \to \mathbb{R}$$
$$V^\pi(s) = \mathbb{E}\left[G_t | s_t = s, \pi\right]$$

Η παραπάνω συνάρτηση μας δίνει την αναμενόμενη τιμή του κέρδους του πράκτορα αν, ξεκινώντας από μία κατάσταση s_t, ακολουθήσουμε την συνάρτηση πολιτικής π. Πρόκειται για μία πραγματική συνάρτηση, δηλαδή για σε κάθε πιθανή κατάσταση αντιστοιχίζει έναν πραγματικό αριθμό. Πέρα από την συνάρτηση V^π, εξαιρετικής σημασίας και ευρείας εφαρμογής είναι η συνάρτηση Q^π (Q-value function) και ορίζεται ως εξής:

$$Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$$
$$Q^\pi(s, \alpha) = \mathbb{E}\left[G_t | s_t = s, \alpha_t = \alpha, \pi\right]$$

Η συνάρτηση Q^π υπολογίζει το αναμενόμενο κέρδος του πράκτορα αν, ξεκινώντας από μία κατάσταση s_t, και εκτελώντας μία δράση α_t ακολουθήσουμε την συνάρτηση πολιτικής π. Στην πραγματικότητα η συνάρτηση αυτή είναι ένας *πίνακας* διαστάσεων $(dim(\mathcal{S}), dim(\mathcal{A}))$, έχει δηλαδή αριθμό γραμμών ίσο με την διάσταση του χώρου των καταστάσεων και αριθμό στηλών ίσο με την διάσταση του χώρου των δράσεων. Σε όλους τους παραπάνω ορισμούς υποθέσαμε ότι η προς προσδιορισμό συνάρτηση πολιτικής είναι μη-ντετερμινιστική, κάτι που στην πραγματικότητα δεν είναι απαραίτητο να ισχύει. Στην περίπτωση που ψάχνουμε για μία ντετερμινιστική συνάρτηση πολιτικής τότε οι δείκτες και οι εκθέτες π αντικαθίστανται από μ.

Ο σκοπός του πράκτορα είναι στο τέλος της MDP που ακολουθεί να έχει μαζέψει το μέγιστο δυνατό κέρδος. Για να το κάνει αυτό θα πρέπει να επιλέγει κάθε φορά την εκάστοτε δράση που θα επιφέρει το μεγαλύτερο κέρδος μετάβασης από μία κατάσταση σε μία άλλη. Πρέπει δηλαδή να "μάθει" την συνάρτηση πολιτικής που μεγιστοποιεί το αναμενόμενο κέρδος. Σε μαθηματικούς όρους, συμβολίζουμε με $Q^*(s, \alpha)$ την *βέλτιστη Q-συνάρτηση κέρδους* (optimal Q-value function):

$$Q^*(s, \alpha) = \max_\pi Q^\pi(s, \alpha)$$

και με $V^*(s)$ την *βέλτιστη V-συνάρτηση κέρδους* (optimal V-value function):

$$V^*(s) = \max_\pi V^\pi(s)$$

Για ένα ζεύγος κατάστασης-δράσης (s, α) η συνάρτηση $Q^*(s, \alpha)$ δίνει το αναμενόμενο κέρδος όταν, όντας στην κατάσταση s λαμβάνουμε την δράση α και έπειτα ακολουθούμε μία βέλτιστη στρατηγική. Με βάση την σκέψη αυτή μπορούμε να εκφράσουμε την συνάρτηση Q^* συναρτήσει της V^* ως εξής:

$$Q^*(s, \alpha) = \mathbb{E}\left[R_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, \alpha_t = \alpha\right] \tag{5.1.1}$$

Στόχος μας είναι να προσδιορίσουμε την *βέλτιστη συνάρτηση πολιτικής*, $\pi^*(s)$, η οποία προφανώς θα ικανοποιεί:

$$\pi^*(s) = \operatorname*{argmax}_{\alpha \in \mathcal{A}} Q^*(s, \alpha) \tag{5.1.2}$$

Οι παραπάνω συναρτήσεις Q^π και π^* είναι πρακτικά αδύνατον να υπολογισθούν αναλυτικά, λόγω της μεγάλης πολυπλοκότητάς τους. Συνεπώς θα πρέπει να βρεθεί ένας τρόπος να γίνει μία προσέγγιση αυτών. Όπως θα δούμε παρακάτω, τον ρόλο αυτόν του συναρτησιακού

προσεγγιστή θα τον παίξει ένα κατάλληλα σχεδιασμένο νευρωνικό δίκτυο. Θα δημιουργήσουμε δηλαδή δύο ξεχωριστά νευρωνικά δίκτυα, ένα που θα μας υπολογίζει την συνάρτηση Q^* και ένα που θα μας υπολογίζει την συνάρτηση π^*.

5.2 Εξισώσεις Βελτιστοποίησης Bellman

Οι εξισώσεις βελτιστοποίησης του Bellman (Bellman Optimality Equations) [16] είναι αναδρομικές εξισώσεις που μας δίνουν έναν τρόπο υπολογισμού των βέλτιστων συναρτήσεων κέρδους, $Q^*(s,\alpha)$ και $V^*(s)$ και φαίνονται παρακάτω:

$$V^*(s) = \max_{\alpha\in\mathcal{A}} \sum_{s'\in\mathcal{S}} \mathcal{P}(s,\alpha,s') \left(\mathcal{R}(s,\alpha,s') + \gamma V^*(s')\right) \tag{5.2.1a}$$

$$Q^*(s,\alpha) = \sum_{s'\in\mathcal{S}} \mathcal{P}(s,\alpha,s') \left(\mathcal{R}(s,\alpha,s') + \gamma \max_{\alpha'\in\mathcal{A}} Q^*(s',\alpha')\right) \tag{5.2.1b}$$

Οι παραπάνω εξισώσεις στηρίζονται στο γεγονός ότι η τιμή μίας κατάστασης υπό την βέλτιστη στρατηγική, $V^*(s)$, θα πρέπει να ισούται με το αναμενόμενο κέρδος για την καλύτερη δράση από αυτήν την κατάσταση, δηλαδή:

$$V^*(s) = \max_{\alpha\in\mathcal{A}(s)} Q^{\pi^*}(s,\alpha)$$

Μία σύντομη απόδειξη των αναδρομικών εξισώσεων Bellman παρατίθεται στο παράρτημα.

Ένα ενδιαφέρον ερώτημα είναι κατά πόσο οι παραπάνω εξισώσεις μπορούν όντως να δώσουν λύση, δηλαδή αν πράγματι υπάρχει μία βέλτιστη λύση. Θα ασχοληθούμε μόνο με την περίπτωση της συνάρτησης $Q^*(s,\alpha)$ όμως παρόμοιες συλλογιστικές πορείες ισχύουν φυσικά και για την $V^*(s)$. Για να απαντήσουμε στο ερώτημα αυτό θεωρούμε τον *συναρτησιακό τελεστή Bellman* $\mathcal{B}$ [15], ο οποίος ορίζεται ως εξής: Αν $K : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ τότε:

$$(\mathcal{B}K)(s,\alpha) = \sum_{s'\in\mathcal{S}} \mathcal{P}(s,\alpha,s') \left(\mathcal{R}(s,\alpha,s') + \gamma \max_{\alpha'\in\mathcal{A}} K(s',\alpha')\right) \tag{5.2.2}$$

Η παραπάνω εξίσωση είναι ακριβώς η εξίσωση βελτιστοποίησης του Bellman για μία συνάρτηση K. Συνεπώς για να αποδείξουμε την ύπαρξη της βέλτισης λύσης, $Q^*(s,\alpha)$, αρκεί να δείξουμε ότι ο τελεστής Bellman είναι μία *συστολή* (contraction), δηλαδή ικανοποιεί την εξής συνθήκη: Για δύο συναρτήσεις $K_1, K_2 : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ ισχύει η ανισότητα:

$$||\mathcal{B}(K_1) - \mathcal{B}(K_2)||_\infty < ||K_1 - K_2||_\infty \tag{5.2.3}$$

Αν καταφέρουμε να δείξουμε ότι ο τελεστής $\mathcal{B}$ είναι συστολή τότε από το *Θεώρημα Σταθερού Σημείου του Banach* θα υπάρχει σταθερό σημείο του τελεστή αυτού και άρα θα υπάρχει μία συνάρτηση που θα ικανοποιεί την εξίσωση βελτιστοποίησης Bellman (η πλήρης

διατύπωση του θεωρήματος αυτού καθώς και η απόδειξή του παρατίθενται στο παράρτημα). Σημειώνεται ότι ο λόγος που επιλέγεται η l^∞ νόρμα στον παραπάνω ορισμό είναι για λόγους ευκολίας πράξεων. Αφού οι χώροι $\mathcal{S}$ και $\mathcal{A}$ είναι πεπερασμένης διάστασης όλες οι νόρμες είναι ισοδύναμες. Έχουμε λοιπόν ότι:

$$||\mathcal{B}(K_1) - \mathcal{B}(K_2)||_\infty =$$

$$\max_{s,\alpha} \left| \sum_{s' \in \mathcal{S}} \mathcal{P}(s,\alpha,s')(\mathcal{R}(s,\alpha,s') + \gamma \max_{\alpha' \in \mathcal{A}} K_1(s',\alpha') - \sum_{s' \in \mathcal{S}} \mathcal{P}(s,\alpha,s')(\mathcal{R}(s,\alpha,s') + \gamma \max_{\alpha' \in \mathcal{A}} K_2(s',\alpha')) \right|$$

$$= \max_{s,\alpha} \left| \sum_{s' \in \mathcal{S}} \mathcal{P}\gamma(\max_{\alpha' \in \mathcal{A}} K_1(s',\alpha') - \max_{\alpha' \in \mathcal{A}} K_2(s',\alpha')) \right|$$

$$\leq \gamma \max_{s,\alpha} \sum_{s' \in \mathcal{S}} |\mathcal{P}| \left| \max_{\alpha' \in \mathcal{A}} K_1(s',\alpha') - \max_{\alpha' \in \mathcal{A}} K_2(s',\alpha') \right| \leq \gamma \max_{s,\alpha} \sum_{s' \in \mathcal{S}} \left| \max_{\alpha' \in \mathcal{A}} K_1(s',\alpha') - \max_{\alpha' \in \mathcal{A}} K_2(s',\alpha') \right|$$

Στην τελευταία ανίσωση χρησιμοποιήθηκε το γεγονός ότι η συνάρτηση $\mathcal{P}$ είναι συνάρτηση πιθανότητας και άρα θα ισχύει ότι $|\mathcal{P}| \leq 1$. Συνεχίζουμε παρατηρώντας ότι η ποσότητα μέσα στην απόλυτη τιμή δεν εξαρτάται πλέον από τα s και α :

$$\gamma \max_{s,\alpha} \sum_{s' \in \mathcal{S}} \left| \max_{\alpha' \in \mathcal{A}} K_1(s',\alpha') - \max_{\alpha' \in \mathcal{A}} K_2(s',\alpha') \right| = \gamma \sum_{s' \in \mathcal{S}} \left| \max_{\alpha' \in \mathcal{A}} K_1(s',\alpha') - \max_{\alpha' \in \mathcal{A}} K_2(s',\alpha') \right| \leq$$

$$\gamma \max_{s' \in \mathcal{S}} \left| \max_{\alpha' \in \mathcal{A}} K_1(s',\alpha') - \max_{\alpha' \in \mathcal{A}} K_2(s',\alpha') \right| = \gamma \max_{s' \in \mathcal{S}, \alpha' \in \mathcal{A}} |K_1 - K_2| = \gamma ||K_1 - K_2||_\infty$$

Τέλος επειδή $0 \leq \gamma < 1$ παίρνουμε την επιθυμητή ανισότητα. Συνεπώς ο τελεστής Bellman έχει σταθερό σημείο και το πρόβλημα υπάρξης της βέλτιστης λύσης $Q^*(s,\alpha)$ λύθηκε.

5.3 On-Policy και Off-Policy Ενισχυτική Μάθηση

Στο σημείο αυτό θα πρέπει να κάνουμε μία μικρή διάκριση στο τί σημαίνει η εκπαίδευση (μέσω ενισχυτικής μάθησης) να είναι on-policy και off-policy. Με απλά λόγια ένας πράκτορας που εκπαιδεύεται με on-policy ενισχυτική μάθηση μαθαίνει για το περιβάλλον ακολουθώντας μία πολιτική π και ταυτόχρονα οι παρατηρήσεις του και οι εμπειρίες του στο περιβάλλον αυτό παράγονται με βάση την ίδια αυτή πολιτική π. Ο πράκτορας δηλαδή μαθαίνει πόσο καλό είναι να κάνει κάτι κάνοντάς το. Εν αντιθέσει, ένας πράκτορας που εκπαιδεύεται με off-policy ενισχυτική μάθηση μαθαίνει το περιβάλλον με την χρήση μίας πολιτικής π_1 αλλά οι εμπειρίες και το κέρδος του πράκτορα παράγονται από μία άλλη πολιτική π_2. Ο πράκτορας δηλαδή μαθαίνει πόσο καλό είναι να κάνει κάτι κάνοντας κάτι άλλο. Η λεγόμενη *Q-εκμάθηση* είναι ένα παράδειγμα off-policy αλγορίθμου [16]:

Αλγόριθμος Q-εκμάθησης
Καθορισμός παραμέτρων αλγορίθμου: βήμα εκπαίδευσης $\tau \in (0, 1]$, $\gamma \in (0, 1]$ και ένα μικρό $\epsilon > 0$
Αρχικοποίηση πίνακα $Q(s, \alpha)$ για κάθε $s \in \mathcal{S}$, $\alpha \in \mathcal{A}$, αυθαίρετα εκτός από $Q(final, \cdot) = 0$
For each episode **do**:
Αρχικοποίηση μίας κατάστασης s
For each time step of episode **do**:
Επιλογή μιας $A \in \mathcal{A}$ από την s μέσω μίας πολιτικής που παράγεται από την Q, έστω π_1
Λήψη της δράσης A, παρατήρηση των s' και R
$Q(s, A) \leftarrow Q(s, A) + \tau[R + \gamma \max_\alpha Q(s', \alpha) - Q(s, A)]$
$s \leftarrow s'$
μέχρι την τερματική κατάσταση s_T

Πίνακας 3: Off-policy Αλγόριθμος Q-εκμάθησης

Παρατηρούμε ότι στον αλγόριθμο Q-εκμάθησης ο πράκτορας μαθαίνει την βέλτιστη στρατηγική (στρατηγική ή πολιτική επικαιροποίησης, update policy) χρησιμοποιώντας την πολιτική $\pi(s) = \max_\alpha Q(s', \alpha)$, η οποία στην βιβλιογραφία αναφέρεται και ως *άπληστη στρατηγική* (greedy policy), όμως συμπεριφέρεται και δρά (επιλέγει δηλαδή την επόμενη κατάσταση) σύμφωνα με την πολιτική π_1 (στρατηγική ή πολιτική συμπεριφοράς, behaviour policy). Ακριβώς επειδή η πολιτική επικαιροποίησης και η πολιτική συμπεριφοράς είναι διαφορετικές ο αλγόριθμος χαρακτηρίζεται ως off-policy. Όταν οι δύο πολιτικές αυτές είναι ίδιες τότε ο αλγόριθμος χαρακτηρίζεται ως on-policy. Ένας τέτοιος αλγόριθμος είναι και ο αλγόριθμος SARSA [16]:

Αλγόριθμος SARSA
Καθορισμός παραμέτρων αλγορίθμου: βήμα εκπαίδευσης $\tau \in (0,1]$, $\gamma \in (0,1]$ και ένα μικρό $\epsilon > 0$
Αρχικοποίηση πίνακα $Q(s,\alpha)$ για κάθε $s \in \mathcal{S}$, $\alpha \in \mathcal{A}$, αυθαίρετα εκτός από $Q(final,\cdot) = 0$
For each episode **do**:
Αρχικοποίηση μίας κατάστασης s
For each time step of episode **do**:
Επιλογή μιας $A \in \mathcal{A}$ από την s μέσω μίας πολιτικής που παράγεται από την Q, έστω π_1
Λήψη της δράσης A, παρατήρηση των s' και R
Επιλογή μιας $A' \in \mathcal{A}$ από την s' μέσω της ίδιας πολιτικής π_1
$Q(s,A) \leftarrow Q(s,A) + \tau[R + \gamma Q(s',A') - Q(s,A)]$
$s \leftarrow s'$ και $A \leftarrow A'$
μέχρι την τερματική κατάσταση s_T

Πίνακας 4: On-policy Αλγόριθμος SARSA

Βλέπουμε λοιπόν ότι η ενισχυτική μάθηση είναι στην ουσία ένα "παιχνίδι", στο οποίο ο παίχτης προσπαθεί να φτάσει σε μία τελική κατάσταση (ουσιαστικά στην νίκη του παιχνιδιού) μαζεύοντας κατά το δυνατόν περισσότερο κέρδος. Φυσικά για έναν άνθρωπο, που μαθαίνει με βιολογικό τρόπο, αυτό το κέρδος δεν έχει κάποιο ιδιαίτερο φυσικό νόημα. Το συσσωρευμένο κέρδος είναι ο τρόπος με τον οποίο βοηθάμε τον υπολογιστή μας να καταλάβει κατά πόσο βρίσκεται στην σωστή πορεία μέσα σε αυτό το παιχνίδι (όπως βοηθάμε το ζώο να καταλάβει κατά πόσο είμαστε ικανοποιημένοι με την συμπεριφορά του δίνοντας του κάτι να φάει). Η ενισχυτική μάθηση βρίσκεται στο ζενίθ της σύγχρονης τεχνητής νοημοσύνης καθώς βρίσκει εφαρμογή σε ποικίλες εφαρμογές, όπως για παράδειγμα στην οδήγηση οχημάτων, στη ρομποτική καθώς και σε πολλά πραγματικά παιχνίδια [17].

6 Ενισχυτική Μάθηση Βάθους (Deep Reinforcement Learning)

6.1 Εισαγωγή και Ιστορική Αναδρομή

Έχοντας τώρα την θεωρία της εκπαίδευσης των νευρωνικών δικτύων και της ενισχυτικής μάθησης μπορούμε να τις συνδυάσουμε με έναν φυσικό τρόπο για να δημιουργήσουμε την θεωρία της *Ενισχυτικής Μάθησης Βάθους*. Όπως έχει ήδη προαναφερθεί, τα νευρωνικά δίκτυα στην ουσία δρουν ως συναρτησιακοί προσεγγιστές, δηλαδή εκφράζουν μία συνάρτηση ως έναν γραμμικό συνδιασμό σιγμοειδών συναρτήσεων. Επίσης θυμόμαστε από την θεωρία της ενισχυτικής μάθησης ότι ο στόχος μας είναι να προσιορίσουμε τις συναρτήσεις $\mu^*(s)$ και $Q^*(s, \alpha)$. Η θεωρία της ενισχυτικής μάθησης βάθους συνδυάζει ακριβώς αυτά τα δύο πράγματα. Προσπαθούμε να προσδιορίσουμε τις συναρτήσεις $\mu^*(s)$ και $Q^*(s, \alpha)$ με νευρωνικά δίκτυα. Ένα νευρωνικό δίκτυο θα προσδιορίζει την συνάρτηση $\mu^*(s)$ και θα λέγεται *δράστης* και ένα άλλο νευρωνικό δίκτυο θα προσδιορίζει την συνάρτηση $Q^*(s, \alpha)$ και θα λέγεται *κριτής*. Η οντότητα που αποτελείται από έναν δράστη και έναν κριτή θα λέγεται *πράκτορας* (agent) και θα αποτελεί στην ουσία τον ρυθμιστή της διεργασίας μας. Πρίν όμως συνεχίσουμε με την μαθηματική θεμελίωση της ενισχυτικής μάθησης βάθους θα κάνουμε μία σύντομη ιστορική αναδρομή που σχετίζεται με την εκπληκτική επίδοση της εφαρμογής της εν λόγω θεωρίας.

Θα είναι λογικό να αναμένει κανείς πως ηλεκτρονικά παιχνίδια, όπως αυτά που έχουν φτιαχτεί για την κονσόλα Atari, αλλά και επιτραπέζια παιχνίδια όπως το σκάκι, θα αποτελούν επιλογή και προνόμιο μόνο ενός ανθρώπου, δηλαδή ενός όντος με *συνείδηση*. Το σκεπτικό αυτό καταρρίφθηκε ήδη από τον Μάιο του 1997, όπου το υπερυπολογιστικό σύστημα Deep Blue της IBM κατάφερε να νικήσει τον παγκόσμιο πρωταθλητή σκακιού, Garry Kasparov. Ο τρόπος με τον οποίο κατάφερε ο υπολογιστής αυτός αυτό το επίτευγμα είναι αξιοποιώντας έναν τεράστιο όγκο δεδομένων από παρτίδες σκακιού, έτσι ώστε να "μάθει" το παιχνίδι και στην συνέχεια, με την βοήθεια ορισμένων αλγορίθμων βελτιστοποίησης (που ανήκουν στον κλάδο της μηχανικής μάθησης), μπόρεσε να "καταλάβει" ποιες κινήσεις και στρατηγικές είναι οι καταλληλότερες για να κερδίσει. Αυτό και μόνο είναι εντυπωσιακό. Ένα ακόμη πιο εντυπωσιακό επίτευγμα ήρθε από την Deep Mind το 2013.

Οι άνθρωποι της Deep Mind κατάφεραν να φτιάξουν ένα σύστημα τεχνητής νοημοσύνης το οποίο μπορούσε να παίξει (μεταξύ άλλων) το ηλεκτρονικό παιχνίδι *Breakout*. Αυτό στην αρχή μπορεί να φαίνεται σαν ένα τίποτα μπροστά στην νίκη του Deep Blue αφού το παιχνίδι *Breakout* είναι, όπως μπορεί κάποιος να επιβεβαιώσει άμεσα παιζοντάς το στο διαδίκτυο, πολύ απλό και εύκολο σε σχέση με το σκάκι. Δεν απαιτεί ούτε ιδιαίτερη νοημοσύνη ούτε πολλές ικανότητες. Όμως η μέθοδος της Deep Mind δεν έπαιξε απλά το παιχνίδι. Στην πραγματικότητα είχαν δημιουργήσει ένα σύστημα τεχνητής νοημοσύνης που δεν γνώριζε *απολύτως τίποτα* για το παιχνίδι, ούτε καν για *έννοιες* όπως παιχνίδι, ρακέτες, τούβλα ή τοίχος [18] (στοιχεία που υπάρχουν στο *Breakout*). Το μόνο "δεδομένο" που διέθετε το σύστημα τεχνητής νοημοσύνης ήταν η βαθμολογία (score) του παιχνιδιού, την οποία είχε ως στόχο να μεγιστοποιήσει. Έπρεπε δηλαδή να μάθει από την αρχή το παιχνίδι χωρίς οδηγίες, απλά παίζοντάς το. Οι αρχικές του επιδόσεις ήταν άθλιες. Καθώς περνούσε όμως

ο χρόνος τα κατάφερνε όλο και καλύτερα μέχρι ωσότου να γίνει κάτι "μαγικό". Το σύστημα αυτό κατάφερε να βρει έναν πανέξυπνο τρόπο για να μεγιστοποιήσει την βαθμολογία του και ο τρόπος αυτός αναδεικνύεται καλύτερα στο βίντεο που βρίσκεται στο τέλος των ηλεκτρονικών παραπομπών. Η μέθοδος που χρησιμοποίησαν δεν ήταν άλλη από την ενισχυτική μάθηση βάθους. Η ίδια μέθοδος έχει διαπρέψει και σε πολλά άλλα ηλεκτρονικά παιχνίδια [17]. Όμως ακόμη και αυτό δεν είναι τίποτα μπροστά στο επόμενο επίτευγμα.

Το επόμενο σύστημα τεχνητής νοημοσύνης της Deep Mind, με όνομα *AlphaGo*, κατάφερε να εκθρονίσει τον παγκόσμιο πρωταθλητή του παιχνιδιού *Go*, Lee Sedol, τον Μάρτιο του 2016 [18]. Το παιχνίδι *Go* είναι ένα παιχνίδι στρατηγικής, όπως το σκάκι, που όμως έχει απίστευτα μεγαλύτερη πολυπλοκότητα. Σε τάξεις μεγέθους, το σκάκι (το οποίο αποτελείται από μία σκακιέρα 8×8 και έχει συνολικά 32 πιόνια και για τους δύο παίχτες) περιλαμβάνει *μόνο* περίπου 10^{43} πιθανούς συνδυασμούς στους οποίους μπορούν να τοποθετηθούν τα πιόνια ακολουθώντας τους κανόνες του παιχνιδιού. Το *Go* (το οποίο παίζεται σε μία σκακιέρα 19×19) περιλαμβάνει περίπου 10^{172} πιθανούς συνδυασμούς! Για να καταλάβουμε πόσο ασύλληπτα μεγάλος είναι αυτός ο αριθμός, έχει υπολογισθεί ότι ο αριθμός των ατόμων στο παρατηρήσιμο σύμπαν είναι της τάξεως των 10^{80}! Πριν τον κρίσιμο αγώνα, όλοι πίστευαν πως μόνο ένας οργανισμός όπως ο άνθρωπος θα μπορέσει ποτέ να παίξει ένα τόσο σύνθετο παιχνίδι. Όταν όμως ο *AlphaGo* νίκησε *τρεις φορές* τον Lee Sedon τα πράγματα άλλαξαν ριζικά. Οι ικανότητες που είχε προσφέρει η ενισχυτική μάθηση βάθους στην επιστήμη της τεχνητής νοημοσύνης ήταν αδιαμφισβήτητες. Ένας υπολογιστής μπόρεσε στην κυριολεξία να *μάθει* ένα τόσο περίπλοκο παιχνίδι, χωρίς να ξέρει απολύτως τίποτα για αυτό απλά παίζοντάς το πολλές φορές.

Κλείνουμε τώρα αυτό το μικρό αλλά πολύ ενδιαφέρον παράθυρο στην ιστορία της τεχνητής νοημοσύνης και συνεχίζουμε με την περιγραφή των μαθηματικών συνιστωσών που συναποτελούν την θεωρία της ενισχυτικής μάθησης βάθους. Καταρχάς ας θυμηθούμε ότι στόχος μας είναι να εκπαιδεύσουμε δύο νευρωνικά δίκτυα που θα προσεγγίζουν τις συναρτήσεις $\mu^*(s)$ και $Q^*(s, \alpha)$ καθώς και ότι με "εκπαίδευση" νευρωνικών δικτύων εννοούμε τον προσδιορισμό των βέλτιστων βαρών και biases. Θα πρέπει λοιπόν τώρα, αν θέλουμε να προσδιορίσουμε τις παραπάνω συναρτήσεις, να καταφέρουμε να τις κάνουμε συναρτήσεις και των παραμέτρων του αντίστοιχου δικτύου που θα τις παράγει.

6.2 Αλγόριθμοι Ντετερμινιστικής Βαθμίδας Πολιτικής (Deterministic Policy Gradient Algorithms)

Οι αλγόριθμοι ντετερμινιστικής βαθμίδας πολιτικής χρησιμοποιούνται κατα κόρον σε προβλήματα ενισχυτικής μάθησης με συνεχή χώρο δράσεων [19]. Η βαθμίδα της ντετερμινιστικής πολιτικής είναι η αναμενόμενη τιμή της βαθμίδας της συνάρτησης που παίρνει ως τιμές τις δράσεις. Η βασική δηλαδή ιδέα είναι να αναπαραστίσουμε την ντετερμινιστική συνάρτηση πολιτικής, $\mu_\theta : \mathcal{S} \to \mathcal{A}$, με μία παραμετρική κατανομή πιθανότητας $\pi_\theta(\alpha|s) = \mathbb{P}[\alpha|s;\theta]$, η οποία στοχαστικά επιλέγει μία δράση α σε μια κατάσταση s βασιζόμενη στο διάνυσμα των παραμέτρων θ. Οι αλγόριθμοι βαθμίδας πολιτικής παίρνουν δείγματα μέσα από αυτήν την κατανομή και προσαρμόζουν το διάνυσμα των παραμέτρων θ προς την κατεύθυνση του

μεγαλύτερου συσσωρευμένου κέρδους. Για να συνεχίσουμε θα πρέπει να τοποθετηθούμε και πάλι σε μία διαδικασία απόφασης κατά Markov. Υποθέτουμε δηλαδή ότι έχουμε δεδομένο έναν (συνεχή) χώρο καταστάσεων $\mathcal{S}$, έναν (συνεχή) χώρο δράσεων $\mathcal{A}$, μία συνάρτηση μετάβασης $\mathcal{P}$ που ικανοποιεί την Μαρκοβιανή ιδιότητα για κάθε αλυσίδα $s_1, \alpha_1, s_2, \alpha_2, \cdots, s_T, \alpha_T$ στον χώρο των δράσεων, μία συνάρτηση κέρδους $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ και έναν παράγοντα προεξόφλησης γ. $\theta \in \mathbb{R}^n$ είναι ένα διάνυσμα n παραμέτρων και $\pi_\theta(\alpha_t|s_t)$ είναι η δεσμευμένη πυκνότητα πιθανότητας στην α_t. Ο πράκτορας χρησιμοποιεί την πολιτική για να αλληλεπιδράσει με το περιβάλλον και δίνει την τροχία $h_{1\to T} = s_1, \alpha_1, R_1, s_2, \alpha_2, R_2, \cdots, s_T, \alpha_T, R_T$. Συμβολίζουμε με $R_t^\gamma = \sum_{k=t}^{\infty} \gamma^{k-t} R(s_k, \alpha_k)$. Οι συναρτήσεις αναμενόμενου κέρδους είναι οι $V^\mu(s) = \mathbb{E}[R_1^\gamma | s_1 = s; \mu]$ και $Q^\mu(s, \alpha) = \mathbb{E}[R_1^\gamma | s_1 = s, \alpha_1 = \alpha; \mu]$. Στόχος του πράκτορα είναι να βρει εκείνη την πολιτική που μεγιστοποιεί το συσσωρευμένο προεξοφλημένο κέρδος, το οποίο συμβολίζουμε με $\mathcal{J}(\mu) = \mathbb{E}[R_1^\gamma | \mu]$. Επίσης συμβολίζουμε την πυκνότητα πιθανότητας στην κατάσταση s' αφού μεταβούμε για t χρονικά βήματα από την κατάσταση s με $p(s \to s', t, \pi)$. Τέλος συμβολίζουμε την προεξοφλημένη κατανομή κατάστασης , $\int_{\mathcal{S}} \sum_{t=1}^{\infty} \gamma^{t-1} p_1(s) p(s \to s', t, \mu) ds$, με $\rho^\mu(s')$ [20], όπου $p_1(s)$ η πυκνότητα πιθανότητας της αρχικής κατάστασης s_1. Με όλους τους παραπάνω συμβολισμούς μπορούμε να εκφράσουμε την αντικειμενική συνάρτησης $\mathcal{J}$ ως εξής [19]:

$$\mathcal{J}(\mu_\theta) = \int_{\mathcal{S}} \rho^\mu(s) R(s, \mu_\theta(s)) ds = \mathbb{E}_{s\sim\rho^\mu}[R(s, \mu_\theta(s))]$$

Τα σύνολα $\mathcal{S}$ και $\mathcal{A}$ θεωρούμε ότι είναι συμπαγή υποσύνολα του ευκλείδιου χώρου. Τώρα είμαστε έτοιμοι να δώσουμε το βασικό θεώρημα των αλγορίθμων ντετερμινιστικής βαθμίδας [19].

Θεώρημα Βαθμίδας Ντετερμινιστικής Πολιτικής : Ας υποθέσουμε ότι οι βαθμίδες $\nabla_\theta \mu_\theta(s)$ και $\nabla_\alpha Q^\mu(s, \alpha)$ υπάρχουν και είναι συνεχείς συναρτήσεις. Τότε η βαθμίδα της συνάρτησης κόστους ως προς το διάνυσμα των παραμέτρων, $\nabla_\theta \mathcal{J}(\mu_\theta)$, δίνεται από την σχέση:

$$\nabla_\theta \mathcal{J}(\mu_\theta) = \int_{\mathcal{S}} \rho^\mu(s) \nabla_\theta \mu_\theta(s) \nabla_\alpha Q^\mu(s, \alpha)|_{\alpha=\mu_\theta(s)} ds = \mathbb{E}_{s\sim\rho^\mu}[\nabla_\theta \mu_\theta(s) \nabla_\alpha Q^\mu(s, \alpha)] \tag{6.2.1}$$

Στην συνέχεια θα χρησιμοποιήσουμε το παραπάνω θεώρημα για να εξάγουμε τους περίφημους Ντετερμινιστικούς Αλγόριθμους Δράστη-Κριτή. Όπως και στην περίπτωση της απλής ενισχυτικής μάθησης, έτσι και εδώ, οι αλγόριθμοι αυτοί διακρίνονται σε on-policy και off-policy.

6.3 Ντετερμινιστικοί Αλγόριθμοι Δράστη-Κριτή (Deterministic Actor-Critic Algorithms)

Οι αλγόριθμοι τύπου δράστη-κριτή χρησιμοποιούνται για την βελτιστοποίηση διαδικασιών απόφασης Markov που χαρακτηρίζονται από παραμετρικοποιημένες οικογένειες συναρτήσεων

πολιτικής (parametrized policies) [21]. Είναι δηλαδή αλγόριθμοι στοχαστικής βαθμίδας στον παραμετρικοποιημένο χώρο του δράστη. Στην πραγματικότητα πρόκειται για δύο παράλληλους αλγόριθμους. Στον έναν αλγόριθμο ο κριτής χρησιμοποιεί *Εκμάθηση Χρονικής Διαφοράς* (Temporal Difference Learning ή TD-Learning), μία έννοια που θα παρουσιασθεί αργότερα, με μία αρχιτεκτονική γραμμικής προσέγγισης για να μάθει την συνάρτηση αναμενόμενης τιμής, $Q(s, \alpha)$, και στον άλλο ο δράστης επικαιροποιείται με βάση την κατεύθυνση μίας προσεγγιστικής βαθμίδας βασισμένης σε πληροφορίες που παρέχονται από τον κριτή.

- On-policy Ντετερμινιστικός Αλγόριθμος Δράστη-Κριτή

Όπως προαναφέρθηκε, ο αλγόριθμος αυτός έχει δύο συνιστώσες. Ο κριτής υπολογίζει τον πίνακα-συνάρτηση $Q(s, \alpha)$ και ο δράστης χρησιμοποιεί την συνάρτηση αυτή για να υλοποιήσει Gradient Ascent της βαθμίδας της Q. Πιο συγκεκριμένα, ο δράστης προσαρμόζει τις παραμέτρους θ (οι οποίες άτυπα προς το παρόν μέσα στο μυαλό μας είναι οι παράμετροι των νευρωνικών δικτύων, δηλαδή τα βάρη και τα biases) της ντετερμινιστικής πολιτικής $\mu_\theta(s)$ μέσω στοχαστικού Gradient Ascent που περιγράφεται από την εξίσωση (6.2.1.). Η μόνη διαφορά με την εξίσωση αυτή είναι ότι στην θέση της ακριβούς συνάρτησης $Q^\mu(s, \alpha)$ μπαίνει μία προσέγγιση αυτής, $Q^w(s, \alpha)$ [19]. Ο κριτής υπολογίζει προσεγγιστικά την $Q^w(s, \alpha) \approx Q^\mu(s, \alpha)$ με την χρήση κάποιου αλγορίθμου όπως η SARSA. Τα βήματα ενός τέτοιου αλγορίθμου φαίνονται παρακάτω [16]:

$$\begin{aligned} \delta_t &= R_t + \gamma Q^w(s_{t+1}, \alpha_{t+1}) - Q^w(s_t, \alpha_t) \\ w_{t+1} &= w_t + \tau_w \delta_t \nabla_w Q^w(s_t, \alpha_t) \\ \theta_{t+1} &= \theta_t + \tau_\theta \nabla_\theta \mu_\theta(s_t) \nabla_\alpha Q^w(s_t, \alpha_t)|_{\alpha=\mu_\theta(s)} \end{aligned}$$

Το διάνυσμα των παραμέτρων w (και όσες παράμετροι φέρουν δείκτη w, όπως ο τ_w) αναφέρεται στην προσεγγιστική συνάρτηση $Q^w(s, \alpha)$. Οι τιμές αυτές χρησιμοποιούνται τέλος για την επικαιροποίηση του διανύσματος των παραμέτρων θ, το οποίο αναφέρεται στην πραγματική επιθυμητή προς προσδιορισμό συνάρτηση $Q^\mu(s, \alpha)$. Τέλος σημειώνεται ότι η ποσότητα δ_t ονομάζεται *Χρονική Διαφορά* (Temporal Difference).

- Off-policy Ντετερμινιστικός Αλγόριθμος Δράστη-Κριτή

Θα εξετάσουμε τώρα την περίπτωση μίας off-policy μεθόδου που μαθαίνει μία ντετερμινιστική πολιτική $\mu_\theta(s)$ (update policy) μέσω μίας αυθαίρετης στοχαστικής πολιτικής $\beta(s, \alpha) \neq \pi_\theta(s, \alpha)$ (behaviour policy) [19]. Στην περίπτωση αυτή η αντικειμενική συνάρτηση θα έχει την εξής μορφή:

$$\mathcal{J}_\beta(\mu_\theta) = \int_{\mathcal{S}} \rho^\beta(s) Q^\mu(s, \mu_\theta(s)) ds$$

Τώρα το Θεώρημα Ντετερμινιστικής Βαθμίδας μας δίνει την εξής προσέγγιση [19]:

$$\nabla_\theta \mathcal{J}_\beta(\mu_\theta) \approx \int_{\mathcal{S}} \rho^\beta(s) \nabla_\theta \mu_\theta(\alpha|s) Q^\mu(s, \alpha) ds = \mathbb{E}_{s\sim\rho^\beta}\left[\nabla_\theta \mu_\theta(s) \nabla_\alpha Q^\mu(s, \alpha)|_{\alpha=\mu_\theta(s)}\right] \tag{6.3.1}$$

Η παραπάνω εξίσωση δίνει την off-policy βαθμίδα ντετερμινιστικής πολιτικής. Είμαστε έτοιμοι τώρα να αναπτύξουμε έναν αλγόριθμο δράστη-κριτή που επικαιροποιεί την πολιτική στην κατεύθυνση της off-policy βαθμίδας της ντετερμινιστικής πολιτικής. Ομοίως με προηγουμένως, αντικαθιστούμε την ακριβής λύση $Q^{\mu}(s,\alpha)$ με μία προσέγγιση $Q^{w}(s,\alpha)$. Ο κριτής υπολογίζει την συνάρτηση $Q^{w}(s,\alpha) \approx Q^{\mu}(s,\alpha)$ από τροχιές που παράγονται από την $\beta(s,\alpha)$ χρησιμοποιώντας έναν κατάλληλο αλγόριθμο. Στο παρακάτω αλγόριθμο ο κριτής χρησιμοποιεί Q-εκμάθηση για να υπολογίσει την συνάρτηση Q^{μ} [16]:

$$\begin{aligned} \delta_t &= R_t + \gamma Q^{w}(s_{t+1}, \mu_\theta(s_{t+1}) - Q^{w}(s_t, \alpha_t) \\ w_{t+1} &= w_t + \tau_w \delta_t \nabla_w Q^{w}(s_t, \alpha_t) \\ \theta_{t+1} &= \theta_t + \tau_\theta \nabla_\theta \mu_\theta(s_t) \nabla_\alpha Q^{w}(s_t, \alpha_t)|_{\alpha=\mu_\theta(s)} \end{aligned}$$

Στην συνέχεια θα κάνουμε μία μικρή συζήτηση όσον αφορά την προσεγγιστική λύση $Q^{w}(s,\alpha)$. Εν γένει, η αντικατάσταση μίας προσέγγισης $Q^{w}(s,\alpha)$ στην ντετερμινιστική βαθμίδα πολιτικής δεν θα ακολουθήσει απαραίτητα την πραγματική βαθμίδα (και στην πραγματικότητα η πορεία που θα ακολουθήσει μπορεί να μην είναι καν πορεία ανάβασης βαθμίδας). Θα πρέπει δηλαδή η προσέγγιση, $Q^{w}(s,\alpha)$, να πληροί τις απαραίτητες προϋποθέσεις ώστε η βαθμίδα $\nabla_\alpha Q^{\mu}(s,\alpha)$ να μπορεί να αντικατασταθεί από την βαθμίδα $\nabla_\alpha Q^{w}(s,\alpha)$ χωρίς να επηρεάζεται η βαθμίδα της ντετερμινιστικής πολιτικής.

Θα λέμε ότι η συναρτησιακή προσέγγιση $Q^{w}(s,\alpha)$ είναι *συμβατή* με την ντετερμινιστική πολιτική [19] όταν ισχύει ότι $\nabla_\theta \mathcal{J}_\beta(\theta) = \mathbb{E}_{s\sim\rho^{\beta}}\left[\nabla_\theta \mu_\theta(s) \nabla_\alpha Q^{w}(s,\alpha)|_{\alpha=\mu_\theta(s)}\right]$, αν πρόκειται για off-policy και $\nabla_\theta \mathcal{J}(\theta) = \mathbb{E}_{s\sim\rho^{\mu}}\left[\nabla_\theta \mu_\theta(s) \nabla_\alpha Q^{w}(s,\alpha)|_{\alpha=\mu_\theta(s)}\right]$ αν πρόκειται για on-policy . Τις προϋποθέσεις για να είναι μία συναρτησιακή προσέγγιση συμβατή με μία ντετερμινιστική πολιτική μας τις δίνει το παρακάτω θεώρημα [19]:

Θεώρημα Συμβατότητας Συναρτησιακής Προσέγγισης: Μία συναρτησιακή προσέγγιση $Q^{w}(s,\alpha)$ είναι συμβατή με μία ντετερμινιστική πολιτική, $\mu_\theta(s,\alpha)$ εάν:

1) $\nabla_\alpha Q^{w}(s,\alpha)|_{\alpha=\mu_\theta(s)} = \nabla_\theta \mu_\theta(s)^{\intercal} w$

2) Το διάνυσμα των παραμέτρων w ελαχιστοποιεί το μέσο τετραγωνικό σφάλμα $MSE(\theta, w) = \mathbb{E}\left[\epsilon(s;\theta,w)^{\intercal}\epsilon(s;\theta,w)\right]$, όπου $\epsilon(s;\theta,w) = \nabla_\alpha Q^{w}(s,\alpha)|_{\alpha=\mu_\theta(s)} - \nabla_\alpha Q^{\mu}(s,\alpha)|_{\alpha=\mu_\theta(s)}$

Το θεώρημα ισχύει τόσο στην on-policy όσο και στην off-policy περίπτωση. Η απόδειξη είναι απλή και παρατίθεται στο παράρτημα.

6.4 Τα Νευρωνικά Δίκτυα του Δράστη και του Κριτή

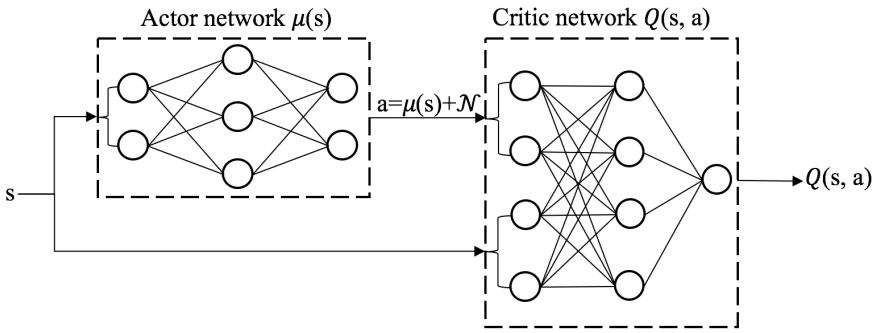
Έχοντας τώρα θεμελιώσει την βασική θεωρία πίσω από τους αλγορίθμους δράστη-κριτή θα πρέπει να βάλουμε στο παιχνίδι και τα νευρωνικά δίκτυα. Ο λόγος που είναι απαραίτητη η εισαγωγή τους είναι επειδή χρειαζόμαστε μία μαθηματική αρχιτεκτονική η οποία θα δράσει ως συναρτησιακός προσεγγιστής τόσο για την ντετερμινιστική συνάρτηση πολιτικής $\mu(s)$

όσο και για την συνάρτηση $Q(s, \alpha)$. Όπως έχει ήδη αναφερθεί στην εισαγωγή αυτού του κεφαλάιου το νευρωνικό δίκτυο που θα υπολογίζει σε κάθε χρονικό βήμα την συνάρτηση πολιτικής θα λέγεται δράστης και το νευρωνικό δίκτυο που θα υπολογίζει την συνάρτηση Q θα λέγεται κριτής. Θα εξετάσουμε τώρα την αρχιτεκτονική του κάθε δικτύου ξεχωριστά.

Δράστης: Είσοδος για το νευρωνικό δίκτυο του δράστη (δηλαδή το στρώμα εισόδου) θα αποτελεί η *τρέχουσα κατάσταση* του συστήματος. Η διάσταση του διανύσματος της κατάστασης θα είναι ίση με τον αριθμό των κόμβων ή νευρώνων του στρώματος εισόδου του δικτύου. Στην συνέχεια η πληροφορία αυτή θα *προωθείται* κατά μήκος του δικτύου με βάση την διαδικασία του feed-forward που περιγράφηκε στο κεφάλαιο **4**. Το διάνυσμα των παραμέτρων του δράστη, δηλαδή τα βάρη και τα biases του θα τα συμβολίσουμε με $\mathbf{W}_\alpha$. Η πληροφορία φτάνει τελικά στο στρώμα εξόδου του δράστη, η οποία θα δίνει ως αποτέλεσμα την *δράση*, α(t). Η διάσταση του διανύσματος της δράσης θα είναι ίση με τον αριθμό των κόμβων ή νευρώνων του στρώματος εξόδου. Με αυτόν τον τρόπο προκύπτει προφανώς μία συνάρτηση, η δράση, η οποία έχει κληρονομήσει όλες τις παραμέτρους του νευρωνικού δικτύου μέσω της προώθησης. Σχηματίστηκε έτσι δηλαδή μία προσέγγιση της συνάρτηση πολιτικής που περιέχει μέσα της ως παραμέτρους τα βάρη του δράστη, $\mu(s; \mathbf{W}_\alpha)$.

Κριτής: Είσοδος για το νευρωνικό δίκτυο του δράστη (δηλαδή το στρώμα εισόδου) θα αποτελεί η τρέχουσα κατάσταση του συστήματος αλλά και η δράση, α(t), που προέκυψε ως έξοδος του νευρωνικού δικτύου του δράστη. Το άθροισμα των διαστάσεων των διανυσμάτων της κατάστασης και της δράσης θα είναι ίσο με τον αριθμό των νευρώνων του στρώματος εισόδου του κριτή. Και εδώ, η πληροφορία προωθείται κατά μήκος του δικτύου μέσω της διαδικασία feed-forward. Το διάνυσμα των παραμέτρων του κριτή, δηλαδή τα βάρη και τα biases του θα τα συμβολίσουμε με $\mathbf{W}_c$. Η έξοδος του δράστη θα είναι η συνάρτηση τιμών Q και θα έχει διαστάσεις $dim(\mathcal{S}) \times dim(\mathcal{A})$. Το γινόμενο των διαστάσεων αυτών θα είναι ίσο με τον αριθμό των νευρώνων του στρώματος εξόδου του κριτή. Και σε αυτήν την περίπτωση έχει σχηματιστεί μία συνάρτηση η οποία έχει κληρονομήσει όλες τις παραμέτρους του κριτή από την διαδικασία της προώθησης. Η συνάρτηση αυτή είναι η $Q(s, \alpha; \mathbf{W}_c)$.

Η σχηματική αναπαραστάση διάδοσης της πληροφορίας δια μέσου των νευρωνικών δικτύων του δράστη και του κριτή μέχρι τον υπολογισμό της συνάρτησης Q φαίνεται στην παρακάτω εικόνα



Εικόνα 7: Σχηματική αναπαράσταση των νευρωνικών δικτύων δράστη και κριτή

Φυσικά έχοντας αποφασίσει ποια θα είναι η δομή των δικτύων μας θα πρέπει να συνεχίζουμε με την εκπαίδευσή τους, δηλαδή τον προσδιορισμό των βέλτιστων παραμέτρων $\mathbf{W}_\alpha$ και $\mathbf{W}_c$. Για τον σκοπό αυτόν θα χρειαστούμε φυσικά ένα κριτήριο βελτιστοποίησης και αυτό θα είναι η *ελαχιστοποίηση μίας αντικειμενικής συνάρτησης* που θα επιβάλλουμε στον κριτή. Οι λεπτομέρειες της συνάρτησης αυτής θα παρουσιασθούν στο επόμενο κεφάλαιο, όπου θα παρουσιασθεί και θα περιγραφεί ο πλήρης αλγόριθμος DDPG. Ο λόγος είναι επειδή η συνάρτηση αυτή δεν περιέχει μονάχα τις παραμέτρους του δράστη και του κριτή αλλά και επιπλέον δύο διανύσματα παραμέτρων που αντιστοιχούν σε άλλα δύο νευρωνικά δίκτυα, τα λεγόμενα *δίκτυα στόχους*.

7 Αλγόριθμος Deep Deterministic Policy Gradient

7.1 Εισαγωγή και Περιγραφή του Αλγορίθμου

Έχοντας τώρα την θεωρία της Ενισχυτικής Μάθησης Βάθους θα παρουσιασθεί και θα επεξηγηθεί ο αλγόριθμος [1, 22, 23] που θα επιλύσει το πρόβλημα της ρύθμισης. Ο πλήρης αλγόριθμος φαίνεται στον παρακάτω πίνακα:

	Αλγόριθμος Εκμάθησης
1	Αρχικοποίηση Δικτύων Δράστη και Κριτή
2	Αρχικοποίηση Δικτύων Στόχου Δράστη και Κριτή
3	Αρχικοποίηση Μνήμης Αναπαραγωγής
4	**For episode = 1 to E do**:
5	Επαναφορά (reset) θορύβου κατά Ornstein-Uhlenbeck $\aleph_t$
6	**For time step = 1 to T do**
7	Παρατήρηση τρέχουσας κατάστασης $s = [y_t, y_{set}]$
8	Εκτέλεση Δράσης $\alpha = u_t = \mu(s, W_\alpha) + \aleph_t$
9	Παρατήρηση επόμενης κατάστασης $s' = [y_{t+1}, y_{set}]$
10	Λήψη αμοιβής r_t
11	Αποθήκευση της τετράδας $[s, \alpha, s', r]$ στην Μνήμη Αναπαραγωγής
12	Λήψη τυχαίας παρτίδας n τετράδων από την Μνήμη Αναπαραγωγής
13	**For minibatch = 1 to** N **do**:
14	Υπολογισμός της ποσότητας $\phi_i = r_i + \gamma Q_t^i$ για κάθε τετράδα της παρτίδας
15	Επικαιροποίηση του Κριτή : $W_c \leftarrow W_c - \alpha \nabla_{\mathbf{W}_c} L(\mathbf{W}_c)$
16	Υπολογισμός του $\nabla_p^i = \frac{\partial Q(s^i, \alpha^i, W_c)}{\partial \alpha^i}$
17	Αντιστροφή της βαθμίδας ∇_p^i
18	Επικαιροποίηση του Δράστη : $W_\alpha \leftarrow W_\alpha + \alpha \nabla_{\mathbf{W}_\alpha} \mathcal{J}(\mathbf{W}_\alpha)$
19	Επικαιροποίηση Δικτύου Στόχου Κριτή: $W_c^t \leftarrow \tau W_c + (1-\tau) W_c^t$
20	Επικαιροποίηση Δικτύου Στόχου Δράστη: $W_\alpha^t \leftarrow \tau W_\alpha + (1-\tau) W_\alpha^t$
21	**endfor**
22	**endfor**
23	**endfor**

Πίνακας 5: Αλγόριθμος εκμάθησης μεθοδολογίας DDPG

Θα προσπαθήσουμε τώρα να εξηγήσουμε βήμα-βήμα τον παραπάνω αλγόριθμο.

Πρώτο βήμα είναι η αρχικοποίηση των νευρωνικών δικτύων του δράστη, του κριτή και των αντίστοιχων δικτύων στόχων (τα δίκτυα στόχοι θα περιγραφούν παρακάτω) με τυχαία βάρη. Τα βάρη αυτά αρχικοποιούνται συνήθως σε μία περιοχή κοντά στο μηδέν για να αποφευχθούν τυχόν προβλήματα υπερχείλησης στα αρχικά στάδια της εκπαίδευσης. Σε αυτό επίσης βοηθά και η τεχνική της κανονικοποίησης κόστους (ενότητα **4.7**). Τα νευρωνικά δίκτυα αυτά θα δράσουν ώς συναρτησιακοί προσεγγιστές, όπως θα δούμε και στην

συνέχεια. Επίσης αρχικοποιείται η μνήμη αναπαραγωγής. Αυτό σημαίνει ότι δημιουργούμε μία άδεια λίστα δεδομένης χωρητικότητας μέσα στην οποία θα αποθηκεύουμε τα αποτελέσματα της εκπαίδευσης.

Έχοντας τώρα έτοιμα τα νευρωνικά δίκτυα και την μνήμη αναπαραγωγής είμαστε έτοιμοι να εξηγήσουμε τις τρεις επαναληπτικές διαδικασίες (for-loops) του αλγορίθμου. Παρατηρούμε ότι η πρώτη επαναληπτική διαδικασία αφορά τα λεγόμενα *επισόδεια*. Το κάθε επισόδειο αποτελείται από *χρονικά βήματα*. Σε κάθε χρονικό βήμα εκτελούνται τα βήματα **7** έως **20** που φαίνονται στον πίνακα. Ο αριθμός των χρονικών βημάτων από τα οποία θα αποτελείται ένα επισόδειο ορίζεται από εμάς, όπως ακριβώς και ο αριθμός των επισοδείων που θα διαρκέσει η εκπαίδευση. Μόλις ο αριθμός των επισοδείων εξαντληθεί ο αλγόριθμος σταματά. Εναλλακτικά ο χρήστης μπορεί να τερματίσει τον αλγόριθμο όταν κρίνει ότι το πρόβλημα της βελτιστοποίησης έχει ολοκληρωθεί. Θα δούμε τώρα τα βήματα του αλγορίθμου που συντελούν ένα επισόδειο.

Αρχικά σχηματίζουμε το διάνυσμα της κατάστασης του συστήματος, **s**. Στον πίνακα φαίνεται ότι αυτό το διάνυσμα αποτελείται από δύο στοιχεία όμως σε μία πραγματική εφαρμογή το διάνυσμα αυτό μπορεί να αποτελείται από μέχρι και δεκάδες στοιχεία. Όσο πιο σύνθετος είναι ο χώρος των πιθανών καταστάσεων (δηλαδή όσο μεγαλύτερη διάσταση έχει) από τόσο περισσότερα στοιχεία θα αποτελείται και το διάνυσμα της κατάστασης. Στην συνέχεια εκτελούμε μία από τις πιθανές δράσεις, **α** και μεταβαίνουμε στην επόμενη κατάσταση, η οποία αναπαριστάται από ένα διάνυσμα **s**' ίδιας διάστασης με την κατάσταση από την οποία προήλθε. Ταυτόχρονα ο πράκτορας λαμβάνει και μία αμοιβή, **r**, η οποία ορίζεται από την συνάρτηση κέρδους που έχει ορίσει ο χρήστης. Μέχρι το σημείο αυτό τα βήματα του αλγορίθμου συντελούν μία διαδικασία απόφασης κατά Markov. Στην συνέχεια τα δεδομένα (**s**, **α**, **s**', **r**) αποθηκεύονται σε μία κενή θέση της μνήμης αναπαραγωγής. Στο σημείο αυτό κρίνεται σκόπιμο να αναφερθεί ότι μία από τις πιο σημαντικές παραμέτρους του αλγορίθμου είναι η συνάρτηση κέρδους $\mathbf{r}(s, \alpha, s')$. Η επιλογή μίας τέτοιας συνάρτησης είναι πολύ λεπτό ζήτημα και μπορεί να κάνει την διαφορά στην εκπαίδευση. Μία συνάρτηση κέρδους μπορεί να οδηγεί σε ταχύτατη και αποδοτική εκπαίδευση ενώ κάποια άλλη ενδέχεται να αποτρέπει τον πράκτορα από το να προσεγγίζει την επιθυμητή κατάσταση.

Στην συνέχεια επιλέγουμε μία τυχαία παρτίδα δεδομένου μεγέθους από διανύσματα της μορφής (**s**, **α**, **s**', **r**) από την μνήμη αναπαραγωγής. Εισερχόμαστε τώρα στην τρίτη και τελευταία επαναληπτική διαδικασία του αλγορίθμου. Για κάθε ένα από αυτά τα διανύσματα που πήραμε από την μνήμη αναπαραγωγής υπολογίζουμε την ποσότητα $\phi_i = r_i + \gamma Q_t^i$. Η ποσότητα αυτή μοιάζει πάρα πολύ με το εσωτερικό του αθροίσματος της εξίσωσης Bellman για την συνάρτηση Q^* και όντως από εκεί πηγάζει η έμπνευση αυτή. Ονομάζεται *στόχος* για λόγους που θα εξηγηθούν στην συνέχεια. Η ποσότητα Q_t^i είναι ένας πίνακας ακριβώς ίδιων διαστάσεων με τον γνωστό από την ενισχυτική μάθηση πίνακα $Q(s, \alpha)$. Παρόλα αυτά ο πίνακας αυτός δεν αποτελεί δεδομένο εξόδου του νευρωνικού δικτύου του κριτή. Αποτελεί έξοδο ενός επιπλέον νευρωνικού δικτύου, του *Κριτή-Στόχου* (Target-Critic). Στο σημείο αυτό θα πρέπει να κάνουμε αλλαγή πορείας και να κάνουμε μία μικρή συζήτηση περί των

δικτύων στόχων [1] που εμφανίζονται στα βήματα **2**, **19** και **20**.

Δίκτυα-Στόχοι (Target Networks): Τα δίκτυα στόχοι έχουν ακριβώς ίδια δομή με αντίστοιχα δίκτυα δράστη και κριτή. Η κρίσιμη διαφορά είναι ότι έχουν διαφορετικές παραμέτρους, δηλαδή διαφορετικά βάρη και biases. Συμβολίζουμε με $\mathbf{W}_\alpha^t$ το διάνυσμα των παραμέτρων του νευρωνικού δικτύου του δράστη-στόχου και με $\mathbf{W}_c^t$ το διάνυσμα των παραμέτρων του κριτή-στόχου. Η εισαγωγή των δικτύων-στόχων στον αλγόριθμο DDPG έχει μία θετική συνέπεια. Κατ'αρχάς καθιστά τον αλγόριθμο *off-policy* [1]. Αυτό γιατί χρησιμοποιούνται διαφορετικά σετ παραμέτρων, τα $\mathbf{W}_\alpha$ και $\mathbf{W}_c$, για τον υπολογισμό των συναρτήσεων $\mu(s)$ και $Q(s,\alpha)$ και διαφορετικά, τα $\mathbf{W}_\alpha^t$ και $\mathbf{W}_c^t$, για την επικαιροποίηση των δικτύων του δράστη και του κριτή. Ο διαχωρισμός αυτός έχει δείξει εμπειρικά ότι καθιστά την εκμάθηση πολύ πιο σταθερή [17]. Ας υποθέσουμε ότι δεν υπήρχαν τα δίκτυα στόχοι. Τότε σε κάθε χρονικό βήμα οι παράμετροι του κριτή, $\mathbf{W}_c$, θα χρησιμοποιούνταν για να αλλάξουν το ίδιο το δίκτυο (και άρα και τις παραμέτρους) του κριτή. Αυτού του είδους επικαιροποίηση μπορεί να προκαλέσει πολύ εύκολα αποκλίσεις, δηλαδή πολύ μεγάλου μέτρου βαθμίδες, κάτι που μπορεί να οδηγήσει σε αποσταθεροποίηση των δικτύων μας. Για τον λόγο αυτόν εισάγουμε τα δίκτυα στόχους. Παρόλα αυτά, ακριβώς επειδή δεν θέλουμε οι παράμετροί τους να ταυτίζονται με τις παραμέτρους των κανονικών δικτύων, η επικαιροποίησή τους γίνεται με πολύ πιο αργό ρυθμό. Για να υλοποιήσουμε αυτήν την χρονική υστέρηση των παραμέτρων των δικτύων-στόχων εισάγουμε άλλη μία παράμετρο η οποία ονομάζεται *ρυθμός εκμάθησης των δικτύων-στόχων* και την συμβολίζουμε με τ. Ο κανόνας επικαιροποίησης των παραμέτρων-στόχων είναι ο εξής:

$$\mathbf{W}_c^t \leftarrow \tau \mathbf{W}_c + (1-\tau)\mathbf{W}_c^t \tag{7.1.1}$$

$$\mathbf{W}_\alpha^t \leftarrow \tau \mathbf{W}_\alpha + (1-\tau)\mathbf{W}_\alpha^t \tag{7.1.2}$$

Φυσικά $0 < \tau \ll 1$. Θα δούμε τώρα πώς αυτές οι παράμετροι-στόχοι χρησιμοποιούνται για να επικαιροποιήσουν τις παραμέτρους του δράστη και του κριτή. Συνεχίζουμε με την περιγραφή του αλγορίθμου.

Τα επόμενα βήματα **15-18** είναι στην ουσία η καρδιά του προβήματος και του αλγορίθμου και αφορούν την επικαιροποίηση των παραμέτρων (βάρη και biases) του κριτή και του δράστη. Όπως είχε αναφερθεί στο κεφάλαιο **7** στους αλγορίθμους τύπου Δράστη-Κριτή ο κριτής ανανεώνει τις παραμέτρους του χρησιμοποιώντας κάποιον αλγόριθμο βελτιστοποίησης, όπως gradient descent, έτσι ώστε να ελαχιστοποιήσει την συνάρτηση κόστους που του έχουμε επιβάλλει και στην συνέχεια ο δράστης χρησιμοποιεί τις καινούριες αυτές παραμέτρους του κριτή έτσι ώστε να ανανεώσει τις δικές του παραμέτρους σύμφωνα με την βαθμίδα της συνάρτησης Q, αυτήν την φορά με gradient ascent, εφόσον στόχος μας είναι η μεγιστοποίηση της συνάρτησης αυτής. Η αντικειμενική συνάρτηση του κριτή είναι η εξής [1, 23]

$$L(\mathbf{W}_c) = \mathbb{E}\left[r + \gamma Q_t^i - Q\right]^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\phi_i - Q(s_i,\alpha_i;\mathbf{W}_c)\right)^2$$

ή

$$L(\mathbf{W}_c) = \frac{1}{N}\sum_{i=1}^{N}\left(r_i(s,\alpha,s') + \gamma Q(s_i', \mu(s_i', \mathbf{W}_\alpha^t); \mathbf{W}_c^t) - Q(s_i, \alpha_i; \mathbf{W}_c)\right)^2 \tag{7.1.3}$$

όπου s' αναφέρεται στην κατάσταση του συστήματος που βρέθηκε ο πράκτορας μετά την λήψη της δράσης α. Αν παρατηρήσουμε τον όρο που βρίσκεται μέσα στο άθροισμα της παραπάνω εξίσωσης θα διαπιστώσουμε ότι πρόκειται για *χρονική διαφορά* (temporal difference), που παρουσιάσθηκε στην ενότητα **6.3**. Η συνάρτηση $\mu(s', \mathbf{W}_\alpha^t)$ είναι η ντετερμινιστική συνάρτηση πολιτικής-στόχος η οποία έχει προκύψει από το νευρωνικό δίκτυο του δράστη-στόχου. Η μόνη διαφορά με το κανονικό δίκτυο του δράστη είναι ότι σαν είσοδος αυτήν την φορά είναι η κατάσταση s' και όχι η κατάσταση s. N είναι το μέγεθος της παρτίδας που λάβαμε από την μνήμη αναπαραγωγής. Ο στόχος ϕ_i είναι σχεδόν ίδιος με το εσωτερικό του αθροίσματος της εξίσωσης Bellman για την συνάρτηση Q. Η κρίσιμη διαφορά είναι ότι εδώ δεν παίρνουμε κάποιο μέγιστο ως προς τις δράσεις. Αυτό συμβαίνει επειδή στα προβλήματα ρύθμισης ο χώρος των δράσεων είναι *συνεχής* οπότε το μέγιστο δεν έχει ιδιαίτερο νόημα [23]. Αντί αυτού σε κάθε χρονικό βήμα του αλγορίθμου κάνουμε μία βελτιστοποίηση ελαχιστοποιώντας την διαφορά μεταξύ της τρέχουσας συνάρτησης $Q(s,\alpha)$ που προέκυψε από τον κριτή και του στόχου, δηλαδή της βέλτιστης λύσης που θα μας έδινε η εξίσωση Bellman αν ο χώρος ήταν διακριτός. Η βαθμίδα της παραπάνω συνάρτησης ως προς τις παραμέτρους του δικτύου είναι η

$$\nabla_{\mathbf{W}_c} L(\mathbf{W}_c) = \mathbb{E}\left[(r + \gamma Q_t - Q(s,\alpha;\mathbf{W}_c))\nabla_{\mathbf{W}_c} Q(s,\alpha,\mathbf{W}_c)\right] \tag{7.1.4}$$

Έπειτα κάνουμε επικαιροποίηση των παραμέτρων (βαρών) του νευρωνικού δικτύου του κριτή με βάση τον αλγόριθμο mini-batch Gradient Descent με την χρήση ενός βελτιστοποιητή (optimizer) του Tensorflow, όπως φαίνεται στο βήμα **15** του αλγορίθμου. Σημειώνεται εδώ ότι ο συμβολισμός $\nabla_{\mathbf{W}_c} L(\mathbf{W}_c)$ σημαίνει μερική παραγώγιση ως προς όλες τις παραμέτρους του δικτύου (βάρη και biases). Συνέχεια έχει η επικαιροποίηση των παραμέτρων του δράστη. Όπως περιγράφηκε στο κεφάλαιο **6** χρησιμοποιούμε τις επικαιροποιημένες παραμέτρους του κριτή έτσι ώστε να υπολογίσουμε την βαθμίδα που δίνεται από την εξίσωση (6.2.1). Αν δηλαδή συμβολίσουμε με $\mathcal{J}(\mathbf{W}_\alpha) = \mathbb{E}\left[Q(s,\alpha;\mathbf{W}_c)\right]$ την αντικειμενική συνάρτηση του δράστη τότε από το Θεώρημα Βαθμίδας Ντετερμινιστικής Πολιτικής [19] παίρνουμε

$$\nabla_{\mathbf{W}_\alpha} \mathcal{J}(\mathbf{W}_\alpha) = \mathbb{E}\left[\nabla_{\mathbf{W}_\alpha}\mu(s,\mathbf{W}_\alpha)\nabla_\alpha Q(s,\alpha;\mathbf{W}_c)|_{\alpha=\mu(s)}\right] \tag{7.1.5}$$

Στην συνέχεια, όπως φαίνεται στο βήμα **18** του αλγορίθμου, οι παράμετροι του δράστη επικαιροποιούνται με βάση τον αλγόριθμο mini-batch gradient ascent, προκειμένου να μεγιστοποιηθεί η συνάρτηση $\mathbb{E}[Q]$, η οποία όπως ξέρουμε από το κεφάλαιο της ενισχυτικής μάθησης είναι άμεσα συνδεδεμένη με το κέρδος του πράκτορα. Τα τελευταία δύο βήματα του αλγορίθμου, **19** και **20**, αποτελούν την επικαιροποίηση των δίκτυων-στόχων. Κλείσαμε έτσι ένα χρονικό βήμα του αλγορίθμου. Η διαδικασία αυτή επαναλαμβάνεται για όσα χρονικά βήματα έχουμε θέσει να διαρκεί ένα επισόδειο και για όσα επισόδεια έχουμε θέσει να διαρκέσει ο αλγόριθμος. Φυσικά ο αλγόριθμος σταματά όταν έχουν εξαντληθεί όλα τα επισόδεια ή όταν ο χρήστης σταματήσει χειροκίνητα τον αλγόριθμο, όταν κρίνει ότι μόνιμη κατάσταση

έχει επιτευχθεί και ότι ο αλγόριθμος έχει συγκλίνει.

Στην συνέχεια θα εξετάσουμε λίγο πιο λεπτομερώς κάποια πιο τεχνικά σημεία του αλγορίθμου, δηλαδή την λειτουργία της μνήμης αναπαραγωγής, τον θόρυβο κατά Ornstein-Uhlenbeck που φαίνεται στα βήματα **5** και **8**, την τεχνική της *κανονικοποίησης παρτίδας*, μίας τεχνικής που χρησιμοποιείται κατά κόρον σε προβλήματα με μεγάλο αριθμό δεδομένων αλλά και σε πολλές εφαρμογές νευρωνικών δικτύων, και την τεχνική της *αντιστροφής βαθμίδας*, δηλαδή το βήμα **17** του αλγορίθμου για το οποίο δεν έγινε καμία αναφορά.

7.2 Μνήμη Αναπαραγωγής (Replay Memory)

Όπως φαίνεται στον πίνακα του αλγορίθμου, σε κάθε time step ενός επισοδείου αποθηκεύουμε την πληροφορία που πήραμε από την λήψη της δράσης στο συγκεκριμένο βήμα στην λεγόμενη *μνήμη αναπαραγωγής* (replay memory). Η πληροφορία αυτή αποτελείται από το τετράνυσμα $(s_t, \alpha_t, s_{t+1}, r_t)$, δηλαδή από την κατάσταση στην οποία βρισκόμαστε στο συγκεκριμένο time step, s_t, την δράση που έλαβε ο δράστης, α_t, την κατάσταση στην οποία βρέθηκε ο πράκτορας μετά την λήψη της δράσης, s_{t+1}, και την αμοιβή που έλαβε ο πράκτορας, r_t. Το τετράνυσμα αυτό καλείται μία *εμπειρία* (experience). Η εμπειρία αυτή που σχηματίζεται σε κάθε time step αποθηκεύεται σε μία εικονική λίστα δεδομένης χωρητικότητας, η οποία στην αρχή του αλγορίθμου αρχικοποιείται άδεια. Όσο η λίστα αυτή γεμίζει με εμπειρίες η εκπαίδευση των νευρωνικών μας δικτύων (δηλαδή τα βήματα του αλγορίθμου που ακολουθούν) διεκπεραιώνεται με βάση αυτές τις υπάρχουσες εμπειρίες. Πιο συγκεκριμένα, σε κάθε time step επιλέγουμε μία τυχαία παρτίδα (mini-batch) συγκεκριμένου μεγέθους εμπειριών από την μνήμη αναπαραγωγής και τις χρησιμοποιούμε για να εκπαιδεύσουμε τα δίκτυά μας. Η τεχνική αυτή έχει ένα μεγάλο πλεονέκτημα σε σύγκριση με το αν χρησιμοποιούσαμε διαδοχικές εμπειρίες (για παράδειγμα τις τελευταίες 64 εμπειρίες) του πράκτορα. Επιλέγοντας τυχαίες εμπειρίες της μνήμης πετυχαίνουμε την *άρση της αυτοσυσχέτισης* μεταξύ διαδοχικών δειγμάτων [24]. Αν τα δίκτυα εκπαιδεύονταν από δείγματα διαδοχικών εμπειριών που προέκυπταν από την εξερεύνηση του περιβάλλοντος τότε τα δείγματα αυτά θα συσχετίζονταν μεταξύ τους όλο και πιο πολύ όσο συνεχιζόταν η εκπαίδευση. Η τυχαιότητα των δειγμάτων σπάει ακριβώς αυτήν την συσχέτιση.

Στο σήμειο αυτό θα πρέπει να αναφερθούν δύο σημαντικές λεπτομέρειες της εν λόγω τεχνικής. Καταρχάς αναφέρθηκε πως σε κάθε time step επιλέγουμε μία τυχαία παρτίδα εμπειριών δεδομένου μεγέθους. Παρόλα αυτά η μνήμη αρχικοποιείται άδεια που σημαίνει ότι στα αρχικά στάδια της εκπαίδευσης δεν υπάρχουν αρκετές εμπειρίες για να πάρουμε δείγμα. Το πρόβλημα αυτό λύνεται εισάγοντας τον λεγόμενο *χρόνο προθέρμανσης* (warm-up time) [25]. Κατά την διάρκεια του χρόνου αυτού *δεν* εκπαιδεύουμε τα νευρωνικά μας δίκτυα αλλά το μόνο που κάνουμε είναι να γεμίζουμε την μνήμη αναπαραγωγής με τυχαίες εμπειρίες. Για να τοποθετηθούμε λίγο καλύτερα στο πρόβλημα, κατά τον χρόνο προθέρμανσης τρέχουμε τον αλγόριθμό μας μέχρι το βήμα **11**, δηλαδή μέχρι ακριβώς πριν την έναρξη της εκπαίδευσης των νευρωνικών δικτύων. Ο χρόνος προθέρμανσης μπορεί να διαρκεί έως και αρκετά επισόδεια και συνήθως τελειώνει μόλις γεμίσει πλήρως η μνήμη αναπαραγωγής.

Με την μνήμη μας γεμάτη εμπειρίες μπορεί να αρχίσει η εκπαίδευσης. Τώρα όμως έχει δημιουργηθεί ένα δεύτερο πρόβλημα που καλούμαστε να λύσουμε. Εφόσον η μνήμη έχει γεμίσει με εμπειρίες τυχαίες, οι οποίες δεν έχουν προκύψει από κάποια λογική συνέχεια ή εξερεύνηση του χώρου (και άρα θα είναι πρακτικά άχρηστες για την εκπαίδευση) και οι παρτίδες που θα επιλέγουμε θα αποτελούνται καθαρά και μόνο από αυτές τις ανούσιες εμπειρίες, αναμένουμε πως η εκπαίδευση θα είναι ανεπιτυχής. Για τον λόγο αυτόν κάνουμε το εξής κόλπο: μόλις γεμίσει πλήρως η μνήμη αναπαραγωγής τότε προκειμένου να βάλουμε την επόμενη εμπειρία στην μνήμη *αφαιρούμε* το πρώτο στοιχείο της μνήμης (αυτό δηλαδή που βρίσκεται στην θέση 0, στο αριστερό άκρο της μνήμης). Με αυτόν τον τρόπο και έχοντας αρχίσει η εκπαίδευση των νευρωνικών δικτύων η μνήμη θα γεμίζει με όλο και καλύτερες εμπειρίες ενώ παράλληλα θα αφαιρούνται οι τυχαίες και άχρηστες εμπειρίες των αρχικών σταδίων του χρόνου προθέρμανσης.

7.3 Θόρυβος Ornstein - Uhlenbeck

Όπως φαίνεται στον κεντρικό αλγόριθμο της μεθοδολογίας DDPG, η δράση α_t αποτελείται από δύο κομμάτια. Το ένα κομμάτι είναι η προς προσδιορισμό συνάρτηση πολιτικής, π. Το δεύτερο κομμάτι αποτελεί τον αποκαλούμενο *Θόρυβο κατά Ornstein - Uhlenbeck*. Για να καταλάβουμε ακριβώς περι τίνος πρόκειται και για ποιόν λόγο χρησιμοποιείται στον αλγόριθμό μας θα πρέπει να εισάγουμε την έννοια της *διαδικασίας κατά Wiener* ή της λεγόμενης *κίνησης κατά Brown*.

Ορισμός: Μία *(μονοδιάστατη) διαδικασία κατά Wiener* (γνωστή και ως *κίνηση κατά Brown*) είναι μία στοχαστική διαδικασία $\{W_t\}_{t\geq 0+}$ με δείκτες τους μη-αρνητικούς πραγματικούς αριθμούς t με τις εξής ιδιότητες:

- $W_0 = 0$
- Η συνάρτηση που ορίζεται από την απεικόνιση $t \to W_t$ είναι συνεχής ως προς την μεταβλητή t με πιθανότητα 1, με την έννοια ότι $\mathbb{P}\left(\limsup_{\epsilon\to 0} \sup_{0\leq s<t\leq 1, t-s\leq\epsilon} \frac{|W(s)-W(t)|}{\sqrt{2\epsilon\log\log(1/\epsilon)}} = 1\right) = 1$
- Η διαδικασία $\{W_t\}_{t\geq 0+}$ έχει ανεξάρτητες από το παρελθόν προσαυξήσεις. Αυτό σημαίνει ότι για κάθε $t > 0$ οι διαφορές $W_{t+u} - W_t$, με $u > 0$ είναι ανεξάρτητες των παλαιότερων τιμών W_s με $s < t$.
- $W_{t+u} - W_t \sim \mathcal{N}(0,\mathrm{u})$.

Η συνάρτηση W_t έχει δύο (μεταξύ άλλων) πολύ ενδιαφέρουσες ιδιότητες. Καταρχάς πρόκειται για μια συνάρτηση που παρόλο που είναι παντού συνεχής δεν είναι πουθενά παραγωγίσιμη. Επιπλέον, αν υποθέσουμε ότι η $\{X_i\}$ είναι μία οικογένεια ανεξάρτητων και ομοίως κατανεμημένων τυχαίων μεταβλητών με μέση τιμή 0 και διασπορά 1, μπορούμε τότε για κάθε n να ορίσουμε την συνεχή στοχαστική διαδικασία:

$$W_n(t) = \frac{1}{\sqrt{n}} \sum_{1 \le k \le \lfloor nt \rfloor} X_i, \qquad t \in [0,1]$$

Η παραπάνω διαδικασία είναι γνωστή και ως *τυχαίος περίπατος*. Εφόσον οι τυχαίες μεταβλητές X_i είναι ανεξάρτητες και οι προσαυξήσεις της W_n θα είναι ανεξάρτητες. Από το κεντρικό οριακό θεώρημα, καθώς $n \to \infty$ έχουμε ότι $W_n(t) - W_n(s) \sim \mathcal{N}(0,$ t - s$)$. Επίσης από το θεώρημα του Donsker καθώς $n \to \infty$ η W_n προσεγγίζει μία διαδικασία κατά Wiener.

Έχοντας την έννοια και την μορφή μίας συνάρτησης τυχαίου περιπάτου είμαστε έτοιμη να δώσουμε την στοχαστική διαφορική εξίσωση που περιγράφει την διαδικασία Ornstein-Uhlenbeck, X_t [26]. Αυτή είναι η παρακάτω:

$$dX_t = -\theta(\mu - X_t)dt + \sigma dW_t \tag{7.3.1}$$

Πρόκειται δηλαδή για μία στοχαστική διαδικασία η οποία καθορίζεται τελικά από έναν τυχαίο περίπατο (κίνηση κατά Brown). Οι παράμετροι θ, μ και σ περιγράφουν το είδος του τυχαίου περιπάτου. Η παράμετρος θ είναι ενδεικτική του πόσο ισχυρά το σύστημά μας αντιδρά σε διαταραχές, η παράμετρος σ δείχνει την διασπορά ή το μέγεθος του θορύβου και η παράμετρος μ είναι ο ασυμπτωτικός μέσος όρος. Η παραπάνω διαφορική εξίσωση λύνεται και παίρνουμε την εξής λύση:

$$X_t = \alpha + (x_0 - \alpha)e^{-\beta t} + \sigma \int_0^t e^{-\beta(t-s)} dW_s \tag{7.3.2}$$

Ο λόγος που προστίθεται ο όρος αυτού του τυχαίου περιπάτου στην δράση α_t είναι επειδή επιτρέπει στον πράκτορα να εξερευνήσει τον (συνεχές) χώρο καταστάσεων με μεγαλύτερη ταχύτητα και απόδοση ακριβώς λόγω της τυχαιότητας που κληρονομεί η δράση από τον θόρυβο αυτόν. Εμπειρικά η εισαγωγή αυτού του είδος θορύβου στην μεθοδολογία του DDPG έχει δείξει ότι αυξάνει τους ρυθμούς εκπαίδευσης του πράκτορα [1].

7.4 Κανονικοποίηση Παρτίδων (Batch Normalization)

Ένα ακόμη τέχνασμα που έχει εμπειρικά δείξει ότι αυξάνει σημαντικά την απόδοση και την ταχύτητα της εκπαίδευσης νευρωνικών δικτύων βάθους είναι αυτό της *κανονικοποίησης παρτίδων*. Για να καταλάβουμε περί τίνος πρόκειται και πώς υλοποιείται θα πρέπει πρώτα να καταλάβουμε ποιο είναι εκείνο το πρόβλημα που καλείται να επιλύσει. Εν γένει η εκπαίδευση νευρωνικών δικτύων με πολλαπλά κρυφά στρώματα παρουσιάζει την εξής δυσκολία: κατά την διάρκεια της εκπαίδευσης οι παράμετροι (δηλαδή τα βάρη και τα biases) αλλάζουν συνεχώς. Αυτό έχει ως αποτέλεσμα η κατανομή των δεδομένων που εισέρχονται σε μία βαθμίδα να αλλάζει και αυτή καθ'όλη την διάρκεια της εκπαίδευσης. Αυτή η συνεχής αλλαγή οδηγεί σε πολύ μικρούς ρυθμούς εκπαίδευσης, εφόσον το κάθε στρώμα θα πρέπει κάθε φορά να προσαρμόζεται σε μία νέα κατανομή και απαιτεί πολύ προσεχτική αρχικοποίηση των παραμέτρων. Το φαινόμενο αυτό καλείται *εσωτερική μεταβλητή μετατόπιση* (internal covariate shift) [27]. Για να λυθεί το συγκεκριμένο πρόβλημα χρησιμοποιούμε έναν αλγόριθμο

κανονικοποίησης των δεδομένων ο οποίος μας εγγυάται ότι τα δεδομένα εισόδου σε κάθε κρυφό στρώμα θα έχουν συνεχώς μία σταθερή κατανομή και συνεπώς θα μειωθεί σημαντικά η εσωτερική μεταβλητή μετατόπιση. Πιο συγκεκριμένα φροντίζουμε έτσι ώστε τα δεδομένα που εισέρχονται σε ένα κρυφό στρώμα να έχουν μέση τιμή 0 και διασπορά 1. Ο μετασχηματισμός που θα πετύχει αυτές τις συνθήκες είναι ο εξής: Αν x είναι η είσοδος ενός νευρώνα ενός κρυφού στρώματος τότε η *κανονικοποιημένη είσοδος*, έστω $\hat{x}$ θα δίνεται από την σχέση:

$$\hat{x} = \frac{x - \mathbb{E}(x)}{Var(x)}$$

Όπως θα δείξουμε σε λίγο οι μετασχηματισμένες αυτές ποσότητες έχουν όντως μέση τιμή ίση με το 0 και διασπορά 1. Επίσης να παρατηρήσουμε ότι αν η είσοδος του στρώματος αποτελείται από ένα n-διάστατο διάνυσμα $x = (x_1, x_2, \ldots, x_n)$, τότε απλώς κανονικοποιούμε την κάθε διάσταση ξεχωριστά:

$$\hat{x}^i = \frac{x^i - \mathbb{E}(x^i)}{Var(x^i)}$$

Στο σημείο αυτό θα πρέπει να παρατηρήσουμε ότι ένας τέτοιος μετασχηματισμός μπορεί να προκαλέσει ένα όχι και τόσο προφανές πρόβλημα. Αν μία κανονικοποιημένη τιμή εισόδου εισέλθει σε μία σιγμοειδή καμπύλη προς επεξεργασία τότε αυτή ωθείται προς την γραμμική περιοχή (σε περιοχές κοντά στο 0), κάτι το οποίο δεν θέλουμε καθώς όπως έχει εξηγηθεί ήδη οι γραμμικές συναρτήσεις δεν είναι κατάλληλες για εκπαίδευση νευρωνικών δικτύων. Για να αντιμετωπίσουμε το πρόβλημα αυτό σε κάθε μεταβλητή x^i αντιστοιχίζουμε ένα ζεύγος παραμέτρων γ^i και β^i και εφαρμόζουμε έναν ακόμα μετασχηματισμό:

$$\hat{x}^i \to \gamma^i \hat{x}^i + \beta^i = \hat{y}^i$$

Οι παράμετροι γ^i και β^i προσδιορίζονται και αυτοί κατά την διάρκεια της εκπαίδευσης, όπως ακριβώς και τα βάρη και τα biases του νευρωνικού δικτύου.

Ας υποθέσουμε τώρα ότι κατά την διάρκεια της εκπαίδευσης χρησιμοποιούμε παρτίδες (mini-batches) δεδομένων που ενδεχομένως να αντλούνται από κάποια μνήμη αναπαραγωγής. Έστω ότι οι παρτίδες αυτές έχουν μέγεθος m. Συνεπώς για έναν νευρώνα ενός κρυφού στρώματος (έστω ότι πρόκειται για τον νευρώνα i) θα έχουμε μία παρτίδα m δεδομένων εισόδου $\mathcal{B} = \{x_k^i\}_{k=1}^m$. Κάθε τέτοια παρτίδα θα έχει μία μέση τιμή, $\mu_\mathcal{B}$ και μία διασπορά $\sigma_\mathcal{B}$ που θα δίνονται από την σχέση:

$$\mu_\mathcal{B} = \frac{1}{m} \sum_{k=1}^{m} x_k^i$$

$$\sigma_\mathcal{B} = \frac{1}{m} \sum_{k=1}^{m} (x_k^i - \mu_\mathcal{B})^2$$

Έχοντας τις παραπάνω τιμές της μέσης τιμής και της διασποράς της παρτίδας εφαρμό-

ζουμε διαδοχικά τους παρακάτω ματασχηματισμούς:

$$x_k^i \to \frac{x_k^i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}} + \epsilon}} = \hat{x}_k^i$$
$$\hat{x}_k^i \to \gamma_k^i \hat{x}_k^i + \beta_k^i = \hat{y}_k^i$$

Ο αριθμός ϵ είναι μία πολύ μικρή παράμετρος ($0 < \epsilon << 1$) και την εισάγουμε μόνο για λόγους αριθμητικής ευστάθειας. Το μόνο που θα πρέπει να δείξουμε τώρα είναι πως πράγματι η μέση τιμή και η διασπορά των μετασχηματισμένων εισόδων $\hat{x}_k^i$ παίρνουν τις επιθυμητές τιμές, δηλαδή 0 και 1 αντίστοιχα. Αρχίζουμε με την μέση τιμή:

$$\mathbb{E}(\hat{x}_k^i) = \mathbb{E}\left(\frac{x_k^i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}} + \epsilon}}\right) = \frac{1}{\sqrt{\sigma_{\mathcal{B}} + \epsilon}}\mathbb{E}(x_k^i - \mu_{\mathcal{B}}) = \frac{1}{\sqrt{\sigma_{\mathcal{B}} + \epsilon}}(\mathbb{E}(x_k^i) - \mathbb{E}(\mu_{\mathcal{B}})) = \frac{1}{\sqrt{\sigma_{\mathcal{B}} + \epsilon}}(\mu_{\mathcal{B}} - \mu_{\mathcal{B}}) = 0$$

$$\sigma(\hat{x}_k^i) = \mathbb{E}[(\hat{x}_k^i)^2] - (\mathbb{E}[\hat{x}_k^i])^2 = \mathbb{E}[(\hat{x}_k^i)^2] = \mathbb{E}\left(\frac{(x_k^i)^2 + (\mu_{\mathcal{B}})^2 - 2x_k^i\mu_{\mathcal{B}}}{\sigma_{\mathcal{B}} + \epsilon}\right) =$$
$$\frac{1}{\sigma_{\mathcal{B}} + \epsilon}\left(\mathbb{E}[(x_k^i)^2] + \mathbb{E}((\mu_{\mathcal{B}})^2) - 2\mu_{\mathcal{B}}\mathbb{E}[x_k^i]\right) = \frac{1}{\sigma_{\mathcal{B}} + \epsilon}\left(\mathbb{E}[(x_k^i)^2] + (\mu_{\mathcal{B}})^2 - 2(\mu_{\mathcal{B}})^2\right) =$$
$$\frac{1}{\sigma_{\mathcal{B}} + \epsilon}\left(\mathbb{E}[(x_k^i)^2] - (\mu_{\mathcal{B}})^2\right) =$$
$$\frac{\sigma_{\mathcal{B}}}{\sigma_{\mathcal{B}} + \epsilon}$$

Έχοντας στο μυαλό μας ότι η παράμετρος ϵ είναι πάρα πολύ μικρή (και για πρακτικούς λόγους μπορεί να αγνοηθεί) παίρνουμε το ζητούμενο.

7.5 Αντιστροφή Βαθμίδας (Gradient Inverter)

Στις περισσότερες περιπτώσεις ρύθμισης μίας διεργασίας ο χώρος των δράσεων είναι ένα φραγμένο σύνολο αφού η κάθε δράση αντιστοιχεί σε μία ενέργεια που βεβαίως έχει μία φυσική σημασία (π.χ. ροή, θερμοκρασία κ.α.). Κατά την διάρκεια της εκπαίδευσης των νευρωνικών δικτύων, ένα αυστηρά μη-κυρτό πρόβλημα βελτιστοποίησης, οι βαθμίδες που υπολογίζονται μπορούν εύκολα να πάρουν πολύ μεγάλες τιμές και συνεπώς οι υπολογιζόμενες δράσεις να βγαίνουν αρκετά εκτός των επιτρεπόμενων ορίων. Για τον λόγο αυτό εισάγεται στον αλγόριθμο το βήμα **17**. Πρόκειται για τον εξής μετασχηματισμό [28]:

$$\nabla_p \to \nabla_p \begin{cases} \frac{p_{max} - p}{p_{max} - p_{min}} & \alpha\nu \ \ \nabla_p > 0 \\ \frac{p - p_{min}}{p_{max} - p_{min}} & \alpha\lambda\lambda\iota\omega\varsigma \end{cases}$$

όπου p_{max} και p_{min} είναι η μέγιστη και η ελάχιστη δράση, που ορίζονται από τον χρήστη. Πρόκειται για έναν μετασχηματισμό κανονικοποίησης της δράσης. Όσο οι τιμές των δράσεων

τείνουν προς τα άκρα του εκάστοτε διαστήματος στο οποίο ορίζονται οι βαθμίδες υποβαθμίζονται και αντιστρέφονται αν οι τιμές αυτές βγούν εκτός των άκρων αυτών. Η μέθοδος της αντιστροφής βαθμίδας δουλεύει με μεγαλύτερη επιτυχία σε μοντέλα όπου οι συναρτήσεις που περιλαμβάνονται είναι λείες (συνεχείς με άπειρης τάξης συνεχείς παραγώγους) και οι κλίσεις που εμφανίζονται δεν είναι ιδιαίτερα μεγάλες. Επίσης κρίνεται σκόπιμο να αναφερθεί ότι το βήμα αυτό του αλγορίθμου είναι προαιρετικό. Ο αρχικός αλγόριθμος της DeepMind που δημιουργήθηκε για να λύσει προβλήματα ρύθμισης δεν περιείχε αυτό το βήμα. Πρόκειται για μία τροποποίηση του αλγορίθμου που μας βοηθά να περιορίσουμε το εύρος των δράσεων.

8 Εφαρμογές και αποτελέσματα

8.1 Εφαρμογή της μεθοδολογίας DDPG σε γραμμικό μοντέλο μίας εισόδου και μίας εξόδου

Ως πρώτο βήμα θα εφαρμόσουμε το προτεινόμενο μοντέλο ρύθμισης σε ένα γραμμικό σύστημα 1×1 το οποίο δανειστήκαμε από την βιβλιογραφία [11]. Η συνάρτηση μεταφοράς που περιγράφει το σύστημα είναι η εξής :

$$G(s) = \frac{0.05}{s^4 - 0.6s^3} \tag{8.1.1}$$

Ο αντίστοιχος χώρος φάσεων περιγράφεται από τις γραμμικές εξισώσεις:

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0.6 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \mathbf{u}$$

$$\mathbf{y} = \begin{bmatrix} 0.05 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}$$

Το φυσικό σύστημα που περιγράφεται από τις παραπάνω εξισώσεις είναι μία διεργασία παραγωγής χαρτιού [11]. Η έξοδος του συστήματος, $\mathbf{y}$(t), είναι η ποσότητα υγρασίας στο χαρτί. Η είσοδος, $\mathbf{u}$, είναι η ροή του ατμού που χρησιμοποιείται για να υγράνει το χαρτί. Σε γλώσσα ενισχυτικής μάθησης ορίζουμε σαν *κατάσταση* (state) του συστήματος να είναι το διάνυσμα:

$$\mathbf{s}(t) = (\mathbf{y}(t), \mathbf{y}_{set})$$

δηλαδή το διάνυσμα που αποτελείται από την τρέχουσα έξοδο του συστήματος και το set point. Ο λόγος που η κατάσταση ορίζεται με αυτόν τον τρόπο είναι επειδή ο πράκτορας (στην ουσία ο ρυθμιστής) πρέπει σε κάθε χρονικό βήμα να γνωρίζει τόσο την τρέχουσα έξοδο του συστήματος όσο και τον τελικό στόχο του (set point) έτσι ώστε να μπορέσει να λάβει την κατάλληλη δράση. Την κατάσταση του συστήματος $\mathbf{y}(t)$ την υπολογίζουμε λύνοντας σε κάθε χρονικό βήμα το παραπάνω γραμμικό σύστημα εξισώσεων με την χρήση της βιβλιοθήκης control της Python. Η βιβλιοθήκη αυτή μας επιτρέπει να κάνουμε πολλές μαθηματικές πράξεις που σχετίζονται με την ρύθμιση διεργασιών. Για παράδειγμα μπορούμε να μετατρέψουμε μία συνάρτηση μεταφοράς σε χώρο φάσεων και αντίστροφα. Επίσης μπορούμε να κάνουμε μία οποιουδήποτε είδους επιβολή στο σύστημά μας και να δούμε την απόκρισή του στο πεδίο του χρόνου.

Η μεταβλητή εισόδου $\mathbf{u}$(t) είναι η *δράση* (action) που θα λαμβάνει ο πράκτορας σε κάθε χρονικό βήμα. Ακριβέστερα, με βάση την μεθοδολογία DDPG, η δράση θα ισούται με:

$$\boldsymbol{\alpha}(t) = \mathbf{u}(t) + \mathcal{N}(t) = \mu(t) + \mathcal{N}(t)$$

όπου $\mu(t)$ η ντετερμινιστική συνάρτηση πολιτικής και $\mathcal{N}(t)$ είναι ο θόρυβος κατά Ornstein-Uhlenbeck. Η λήψη μίας δράσης έχει ως αποτέλεσμα την μετάβαση του πράκτορα από την τρέχουσα κατάσταση στην επόμενη, την οποία θα συμβολίσουμε ως:

$$\mathbf{s}'(t) = (\mathbf{y}(t+1), \mathbf{y}_{set})$$

Στόχος μας είναι, ξεκινώντας από μηδενική τιμή υγρασίας να προσδιορίσουμε εκείνο το προφίλ της ροής του ατμού που θα φέρει το χαρτί μας σε μία κατάσταση επιθυμητής υγρασίας. Το μοντέλο αυτό το δανειστήκαμε από σχετική βιβλιογραφία. Δυστυχώς όμως στην βιβλιογραφία δεν παρατίθενται οι μονάδες μέτρησης τόσο της υγρασίας του χαρτιού όσο και της ροής του ατμού. Για τον λόγο αυτό θα φερθούμε στο σύστημα αυτό ως ένα αμιγώς μαθηματικό μοντέλο που θα αποτελέσει μία αρχική δοκιμασία (benchmark problem) της προτεινόμενης μεθοδολογίας.

Παρακάτω παρουσιάζονται διάφοροι παράμετροι του αλγορίθμου που επιλέχθηκαν για την προσομοίωση:

Η προσομοίωση υλοποιήθηκε σε γλώσσα προγραμματισμού Python, με την βοήθεια (μεταξύ άλλων) της βιβλιοθήκης ανοιχτού κώδικα Tensorflow. Το Tensorflow σχεδιάστηκε από την ομάδα της Google, Google Brain, με σκοπό να κάνει την υλοποίηση πολλών αλγορίθμων μηχανικής μάθησης πιο προσιτή και εύκολη για το ευρή κοινό. Η μεγαλύτερη επιτυχία του Tensorflow βρίσκεται στην εφαρμογή του στα νευρωνικά δίκτυα. Ο λόγος έγκειται στο γεγονός ότι το Tensorflow λειτουργεί δημιουργώντας γραφήματα ροής πληροφορίας, ό,τι ακριβώς είναι και ένα νευρωνικό δίκτυο. Μπορούμε, για παράδειγμα, με μία απλή εντολή να σχεδιάσουμε ένα στρώμα ενός νευρωνικού δικτύου που να περιέχει όσους νευρώνες επιθυμούμε. Μπορούμε επίσης να ορίσουμε ο κάθε νευρώνας ενός στρώματος του δικτύου να συνδέεται με όλους τους νευρώνες του επόμενου κρυφού στρώματος και σε κάθε τέτοια σύνδεση να ορίζεται το αντίστοιχο βάρος. Τέτοιου είδους νευρωνικά δίκτυα λέγονται *πλήρως συνδεδεμένα* (fully-connected). Το Tensorflow κάνει επίσης την διαδικασία του backpropagation τετριμμένη. Οι λεγόμενοι ενσωματωμένοι *βελτιστοποιητές* (optimizers) του Tensorflow μας επιτρέπουν με μεγάλη ευκολία να ελαχιστοποιήσουμε την επιθυμητή συνάρτηση κόστους που εμείς έχουμε ορίσει, εφαρμόζοντας κάποιον αλγόριθμο Gradient Descent (είτε stochastic είτε mini-batch είτε τον κλάσσικό αλγόριθμο). Το όνομα του Tensorflow προέρχεται από τον τρόπο με τον οποίο λειτουργεί σε θεμελιώδες επίπεδο. Συγκεκριμένα, το Tensorflow λειτουργεί χρησιμοποιώντας *τανυστές* (tensors). Οι τανυστές είναι μαθηματικά αντικείμενα που γενικεύουν την έννοια του διανύσματος και του πίνακα σε περισσότερες διαστάσεις (που όμως μετασχηματίζονται με έναν συγκεκριμένο τρόπο όταν αλλάζουμε συντεταγμένες). Αυτοί οι τανυστές μπορεί να αποτελούν δεδομένα εισόδου ή και αποτέλεσμα κάποιας πράξης μέσα στο γράφημα. Η πληροφορία ενός τανυστή *ρέει* (flows) μέσα στο γράφημα μέσω των βαρών και των συναρτήσεων ενεργοποίησης, που μπορούν επίσης να ορισθούν μέσω του Tensorflow. Όλα τα (fully-connected) νευρωνικά δίκτυα, οι βελτιστοποιητές αλλά και άλλες συνιστώσες της προσομοίωσης, όπως η βαθμίδα του κριτή (7.1.5), η κανονικοποίηση παρτίδων και η αντιστροφή βαθμίδας υλοποιήθηκαν με την βοήθεια του Tensorflow.

Ο κώδικας που χρησιμοποιήθηκε είναι στο ίδιο στυλ με αυτόν που υπάρχει στην πρώτη ηλεκτρονική παραπομπή στο τέλος του κειμένου. Όσον αφορά τα νευρωνικά δίκτυα του δράστη, του κριτή αλλά και τα αντίστοιχα δίκτυα-στόχους και τα τέσσερα αποτελούνται από

δύο κρυφά στρώματα. Το πρώτα κρυφό στρώμα έχει 400 κόμβους (νευρώνες) και το δεύτερο 300.Όλα τα στρώματα του κάθε νευρωνικού δικτύού είναι πλήρως συνδεδεμένα Όλες οι συναρτήσεις ενεργοποίησης επιλέχθηκαν να είναι της μορφής ReLU. Ο ρυθμός εκμάθησης είναι κοινός για όλα τα δίκτυα και είναι ίσος με $5 \cdot 10^{-4}$. Τα δίκτυα-στόχοι ανανεώνονται με βάση τα βήματα **19** και **20** με $\tau = 10^{-3}$. Σε όλες τις παραμέτρους του νευρωνικού δικτύου του κριτή επιβάλλουμαι και ένα επιπλέον $\mathcal{L}^2$-κόστος (με πολλαπλασιαστική σταθερά ίση με 0.5). Στην δράση προστέθηκε και θόρυβος κατά Ornstein-Uhlenbeck με σταθερές $\mu = 0, \theta = 0.15, \sigma = 0.2$. Ο παράγοντας προεξόφλησης γ τίθεται ίσος με 0.99. Στην μνήμη αναπαραγωγής δόθηκε χωρητικότητα ίση με 10^4. Ο χρόνος προθέρμανσης είναι ίσος με 10^4 χρονικά βήματα (όπου το κάθε χρονικό βήμα ισούται με μία μονάδα χρόνου), δηλαδή τόσο όσο χρειάζεται για να γεμίσει η μνήμη αναπαραγωγής. Το μέγεθος των παρτίδων είναι ίσο με 64. Τα κρυφά στρώματα είναι κανονικοποιημένα, με βάση την διαδικασία που περιγράφηκε στην ενότητα **7.4**. Χρησιμοποιήθηκε το βήμα **17** της αντιστροφής βαθμίδας και η δράση περιορίστηκε μεταξύ των τιμών $p_{max} = \alpha_{max} = 400$ και $p_{min} = \alpha_{min} = 100$. Σαν βελτιστοποιητής που θα επιχειρήσει να βρεί ελάχιστο της αντικειμενικής συνάρτησης χρησιμοποιήθηκε ο GradientDescentOptimizer του Tensorflow.

Ένα επισόδειο τίθεται να διαρκεί 300 χρονικά βήματα (time steps). Ένα επισόδειο τελειώνει όταν πληρείται μία από τις εξής προϋποθέσεις:

- Το επισόδειο έχει ολοκληρωθεί, δηλαδή ο αλγόριθμος έχει τρέξει για 300 χρονικά βήματα
- Ο ρυθμιστής έχει καταφέρει να εντοπίσει το set point και έχει καταφέρει να παραμείνει ικανοποιητικά κοντά σε αυτό για 5 χρονικά βήματα, δηλαδή $|s' - s_{set}| < eps$, για κάποιο eps που του ορίζουμε εμείς. Εδώ έχουμε θέσει $eps = 0.01$.

Επίσης σημειώνεται ότι στην αρχή του κάθε επισοδείου θα πρέπει να επαναφέρουμε το σύστημά μας στην αρχική του κατάσταση (στην αρχική υγρασία) έτσι ώστε ο ρυθμιστής να αρχίσει εκ νέου την προσπάθειά του να ρυθμίσει το σύστημα.

Στο πρόβλημα αυτό της ρύθμισης συστήματος μίας εξόδου και μίας εισόδου θα ελέγξουμε κατά πόσο ο ρυθμιστής μας μπορεί να προσεγγίσει διαφορετικά set points και πώς επηρεάζει την εκπαίδευση η συνάρτηση κέρδους (reward function) που ορίζουμε. Πιο συγκεκριμένα, στην παρούσα προσομοίωση θα χρησιμοποιήσουμε δύο διαφορετικές συναρτήσεις κέρδους που φαίνονται παρακάτω:

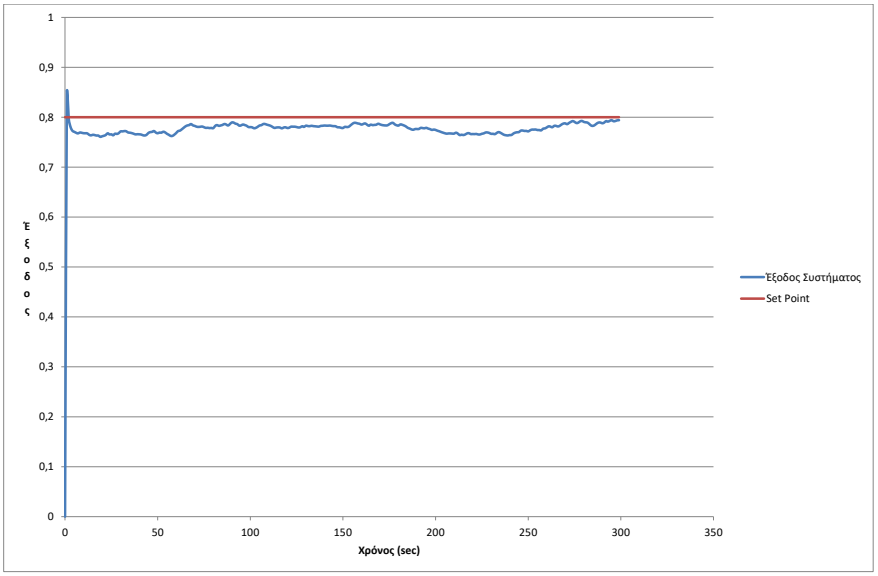
$$\mathcal{R} = \begin{cases} \frac{0.001}{|s' - s_{set}|} & , |s' - s_{set}| < |s - s_{set}| \\ -10 & , |s' - s_{set}| \geq |s - s_{set}| \end{cases}$$

$$\mathcal{R}' = \begin{cases} \frac{0.001}{|s' - s_{set}|} & , |s' - s_{set}| < |s - s_{set}| \\ -10 & , |s' - s_{set}| \geq |s - s_{set}| \\ 100 & |s' - s_{set}| < 0.01 \end{cases}$$

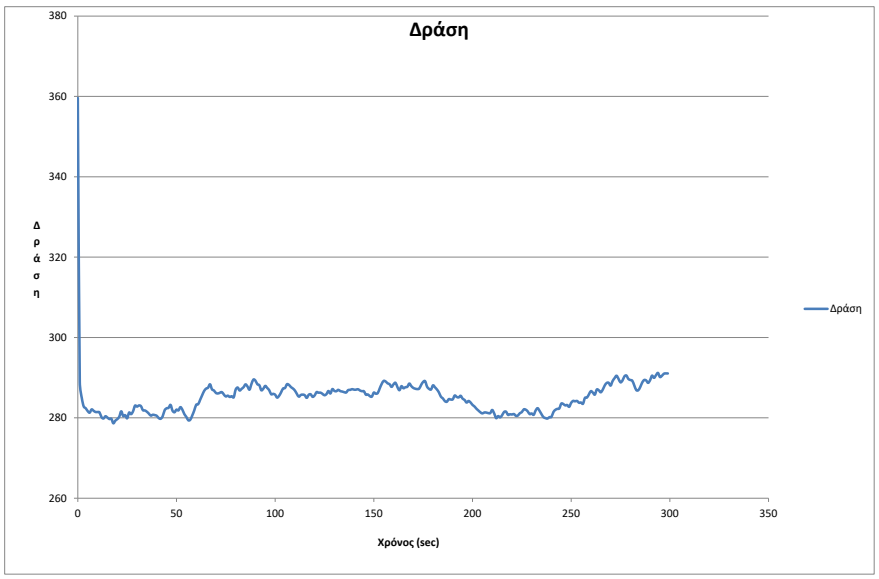
Αρχίζουμε τώρα με την παρουσίαση των αποτελεσμάτων των προσομοιώσεων.

- Set Point = 0.8 και χρήση της $\mathcal{R}$ συνάρτησης κέρδους

Όπως είναι προφανές από τον ορισμό της συνάρτησης κέρδους $\mathcal{R}$, ο πράκτορας λαμβάνει κέρδος ίσο με την μονάδα όταν και μόνο όταν η λήψη της εκάστοτε δράσης έχει ως αποτέλεσμα η κατάσταση στην οποία καταφθάνει ο πράκτορας να απέχει μικρότερη απόσταση από το set point σε σχέση με πριν ληφθεί η δράση. Αν δεν ισχύει αυτό, δηλαδή η δράση οδηγεί τον πράκτορα πιο μακριά από το από το set point σε σχέση με πριν τότε ο πράκτορας χάνει κέρδος ίσο με 10 (δηλαδή η αμοιβή είναι ίση με -10). Η εκπαίδευση διεξήχθη με την χρήση των προαναφερόμενων παραμέτρων και τα αποτελέσματα τόσο για την έξοδο όσο και για την δράση είναι τα εξής:



Εικόνα 8: Προφίλ της εξόδου του συστήματος συναρτήσει του χρόνου με την χρήση της $\mathcal{R}$ συνάρτησης κέρδους



Εικόνα 9: Προφίλ της δράσης του συστήματος συναρτήσει του χρόνου με την χρήση της $\mathcal{R}$ συνάρτησης κέρδους

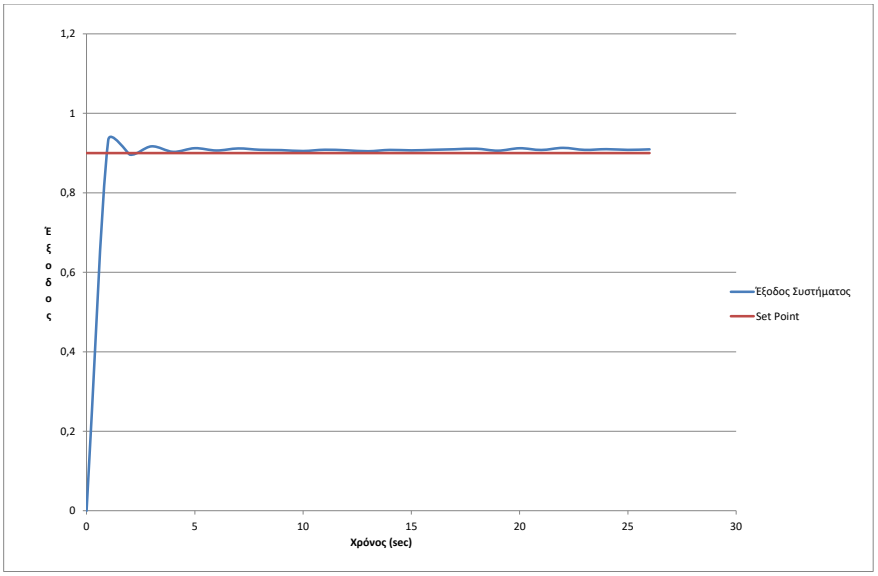
Σε ό,τι αφορά την έξοδο του συστήματος παρατηρούμε ότι σε πολύ μικρό χρονικό διάστημα (σε μόλις ένα χρονικό βήμα) ο πράκτορας καταφέρνει και φτάνει στο set point, το ξεπερνά όμως κάνοντας μία μικρή υπέρβαση. Εν συνεχεία, πέφτει κάτω από το set point και συνεχίζει να το προσεγγίζει με ικανοποιητική ακρίβεια μέχρι το τέλος του επισοδείου οπού και φαίνεται να συγκλίνει σε αυτό. Το προφίλ της εξόδου είναι αρκετά θορυβώδες σε συμπεριφορά και αυτό οφείλεται στο προφίλ της δράσης του πράκτορα. Όπως φαίνεται στο παραπάνω διάγραμμα της δράσης, στα πολύ αρχικά στάδια του επισοδείου, ο πράκτορας δίνει μία αρκετά μεγάλου μεγέθους δράση, η οποία πέφτει με πολύ μεγάλο ρυθμό. Αυτή η αρχική δράση είναι υπεύθυνη για την μικρή υπέρβαση της εξόδου. Αφού η έξοδος υπερβεί το set point, ο πράκτορας ξέρει ότι πρέπει να επιφέρει ραγδαία μείωση της δράσης για να πέσει κάτω από το set point και ύστερα να την προσαρμόσει κατάλληλα για να παραμείνει όσο το δυνατόν πιο κοντά σε αυτό. Και πράγματι μετά την αρχή του επισοδείου (όταν πλέον ο πράκτορας έχει πέσει κάτω από το set point) παρατηρούμε ότι το προφίλ της εξόδου και της δράσης είναι παρόμοια. Στις χρονικές στιγμές που η έξοδος φαίνεται να πλησιάζει το set point η δράση αυξάνεται, ενώ όταν η έξοδος φαίνεται να απομακρύνεται από το set point η δράση μειώνεται. Τελικά συμπεραίνουμε ότι η μόνιμη κατάσταση δεν επιτυγχάνεται κατά την διάρκεια του επισοδείου παρά μόνο ίσως στα τελικά στάδιά του όπου ο πράκτορας φαίνεται να έχει πλησιάσει πολύ κοντά στο set point.

Στην συνέχεια θα εξετάσουμε άλλα δύο διαφορετικά set points με την χρήση όμως αυτήν

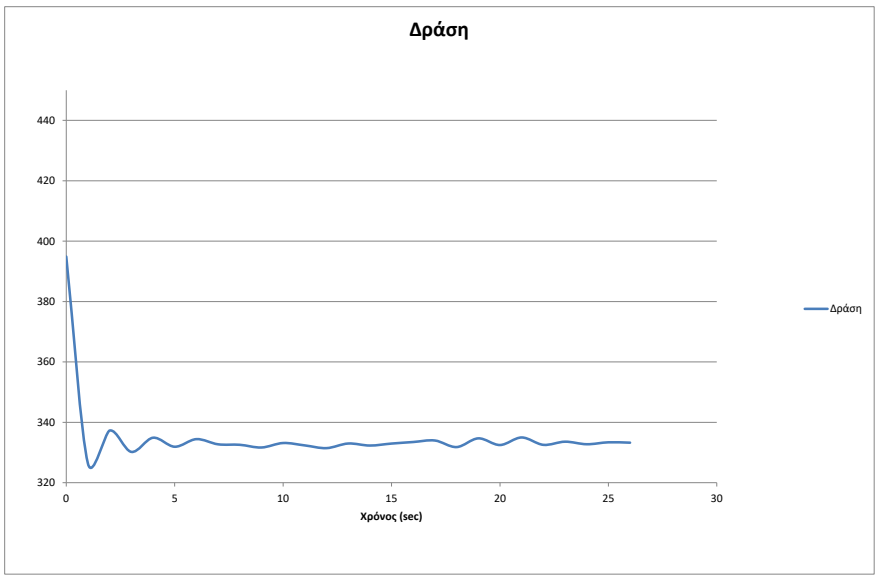
την φορά της συνάρτησης κέρδους $\mathcal{R}'$.

- Set Point = 0.9 και χρήση της $\mathcal{R}'$ συνάρτησης κέρδους

Στην περίπτωση αυτή η συνάρτηση κέρδους $\mathcal{R}'$ έχει μία βασική διαφορά σε σχέση με την $\mathcal{R}$. Η $\mathcal{R}'$ δίνει στον πράκτορα μία πολύ μεγάλη αμοιβή όταν καταφέρνει να είναι αρκετά ικανοποιητικά κοντά στο set point. Εδώ έχουμε θέσει ότι για να λάβει αυτή την μεγάλη αμοιβή θα πρέπει να είναι λιγότερο από 0.01 μακριά από το set point. Η λογική της συνάρτησης αυτής είναι ότι κατα πάσα πιθανότητα θα ωθήσει τον πράκτορα πιο κοντά στο set point και για περισσότερο χρόνο, αφού ο πρωταρχικός σκοπός του είναι να μαζέψει, μέχρι το τέλος του επισοδείου, το μεγαλύτερο δυνατό κέρδος. Τα αποτελέσματα είναι τα εξής:



Εικόνα 10: Προφίλ της εξόδου του συστήματος συναρτήσει του χρόνου με την χρήση της $\mathcal{R}'$ συνάρτησης κέρδους



Εικόνα 11: Προφίλ της δράσης του συστήματος συναρτήσει του χρόνου με την χρήση της $\mathcal{R}'$ συνάρτησης κέρδους

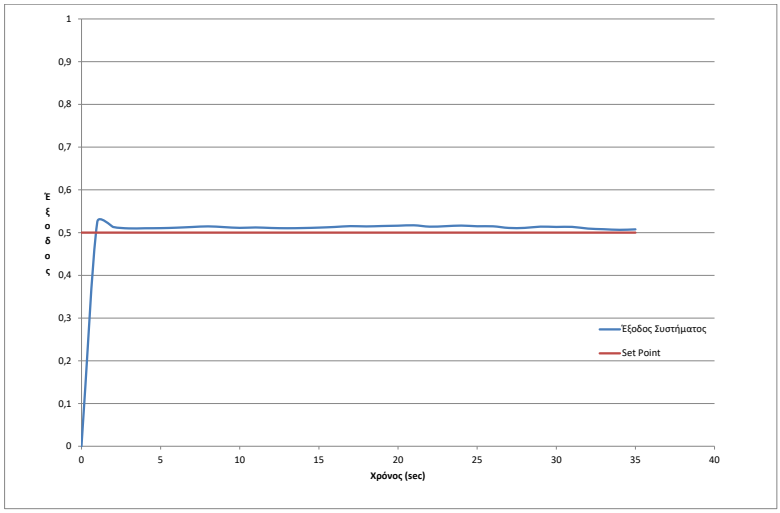
Όπως είναι προφανές τα αποτελέσματα σε αυτήν την περίπτωση είναι αρκετά διαφορετικά και μάλιστα πολύ καλύτερα. Στην αρχή του επισοδείου ο ρυθμιστής επιλέγει να δώσει μία αρκετά μεγάλη δράση η οποία, όπως και στην προηγούμενη περίπτωση, οδηγεί σε μία μικρή υπέρβαση της εξόδου (περίπου της ίδιας τάξεως). Στην συνέχεια η δράση πέφτει με αρκετά μικρότερη κλίση, έτσι ώστε ο πράκτορας να πλησιάσει το set point. Από εκεί και πέρα παρατηρούμε ένα σχετικά ομαλό προφίλ δράσης το οποίο φυσικά συνεπάγεται ένα πολύ ομαλό προφίλ εξόδου. Πιο συγκεκριμένα η έξοδος του συστήματος φαίνεται να ταλαντεύεται πολύ κοντά γύρω από το set point μέχρι το όγδοο χρονικό βήμα και μετά πρακτικά συγκλίνει σε αυτό. Η νέα επιβαλλόμενη συνάρτηση κέρδους όντως ωθεί τον πράκτορα πολύ κοντά στο set point και διατηρεί πολύ κοντινές αποστάσεις γύρω από αυτό. Με αυτόν τον τρόπο ο πράκτορας μπορεί, μέχρι το τέλος του επισοδείου, να μαζέψει αρκετά μεγάλο κέρδος. Η μόνιμη κατάσταση φαίνεται να επιτυγχάνεται με μεγάλη ακρίβεια μετά το δέκατο χρονικό βήμα.

Ως τελευταία εφαρμογή της μεθοδολογίας DDPG σε γραμμικό σύστημα δοκιμάζουμε την ίδια συνάρτηση κέρδους, $\mathcal{R}'$ σε ένα διαφορετικό set point.

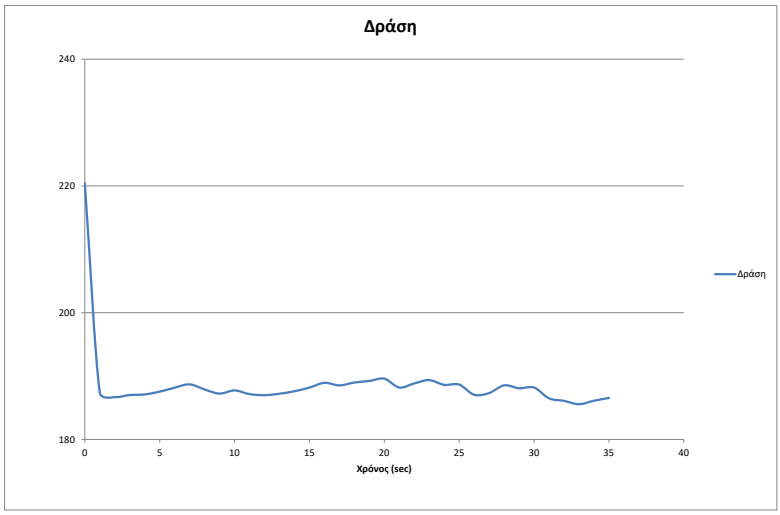
- Set Point $= 0.5$ και χρήση της $\mathcal{R}'$ συνάρτησης κέρδους

Η προσομοίωση αυτή γίνεται πιο πολύ για να δοκιμασθεί η ευρωστία της συνάρτηση κέρδους

$\mathcal{R}'$, δηλαδή για να δούμε αν μπορεί να δώσει τα ίδια επιτυχημένα αποτελέσματα για ένα διαφορετικό set point. Όλες οι υπόλοιπες παράμετροι της προσομοίωσης παραμένουν ίδιες. Παίρνουμε τα εξής αποτελέσματα:



Εικόνα 12: Προφίλ της εξόδου του συστήματος συναρτήσει του χρόνου με την χρήση της $\mathcal{R}'$ συνάρτησης κέρδους



Εικόνα 13: Προφίλ της δράσης του συστήματος συναρτήσει του χρόνου με την χρήση της $\mathcal{R}'$ συνάρτησης κέρδους

Όπως φαίνεται η συμπεριφορά του συστήματος υπό την επίδραση της ίδιας συνάρτησης κέρδους είναι πολύ παρόμοια. Υπάρχει μία μικρή υπέρβαση στα αρχικά στάδια του επισοδείου και έπειτα ο πράκτορας πλησιάζει το set point και παραμένει πολύ κοντά σε αυτό, επιτυγχάνοντας έτσι μόνιμη κατάσταση. Το προφίλ της δράσης είναι προφανώς πολύ παρόμοιο με αυτό που αντιστοιχούσε στο set point 0.9 και είναι αυτό ουσιασικά που καθορίζει και την εξίσου παρόμοια με πριν συμπεριφορά της εξόδου. Συμπεραίνουμε λοιπόν ότι η προτεινόμενη συνάρτηση κέρδους $\mathcal{R}'$ είναι καταλληλότερη για την εκπαίδευση του εν λόγω συστήματος.

Στην συνέχεια θα εξετάσουμε κατά πόσο η μεθοδολογία DDPG μπορεί να αντιμετωπίσει ένα αρκετά πιο δύσκολο πρόβλημα ρύθμισης.

8.2 Εφαρμογή της μεθοδολογίας DDPG σε μη-γραμμικό μοντέλο δύο εισόδων και δύο εξόδων

Το επόμενο βήμα είναι η δοκιμασία της προτεινόμενης μεθοδολογίας ρύθμισης σε ένα μη-γραμμικό μοντέλο, το οποίο δανειστήκαμε από την βιβλιογραφία [29]. Το μοντέλο αυτό δίνεται από τις παρακάτω εξισώσεις:

$$\begin{aligned} \frac{dC_A}{dt} &= \frac{F}{V}(C_{A,in} - C_A) - 2k_0 \exp\left(-\frac{E}{RT}\right) C_A^2 \\ \frac{dT}{dt} &= \frac{F}{V}(T_{in} - T) - 2\frac{(-\Delta H_R)}{\rho c_p} k_0 \exp\left(-\frac{E}{RT}\right) C_A^2 - \frac{UA}{V\rho c_p}(T - T_j) \end{aligned} \tag{8.2.1}$$

Πρόκειται δηλαδή για τις εξισώσεις λειτουργίας (ισοζύγιο μάζας και ενέργειας) ενός *μη-ισοθερμοκρασιακού αντιδραστήρα CSTR* στον οποίο λαμβάνει χώρα η εξώθερμη αντίδραση $A \rightarrow B$ στην υγρή φάση. E είναι η ενέργεια ενεργοποίησης της αντίδρασης, k_0 η σταθερά Arrhenius, R η παγκόσμια σταθερά των αερίων, ΔH_R ο θερμοτονισμός της αντίδρασης, ρ η πυκνότητα της υγρής φάσης που εισέρχεται στον αντιδραστήρα, V ο όγκος του αντιδραστήρα, c_p η θερμοχωρητικότητα του ψυκτικού μέσου, U ο συντελεστής μεταφοράς θερμότητας του ψυκτικού και A η επιφάνεια εναλλαγής θερμότητας.

Στο πρόβλημα αυτό θεωρούμε ως μεταβλητές εισόδου την ροή εισόδου στον αντιδραστήρα, F και την θερμοκρασία του ψυκτικού μέσου, T_j, και ως μεταβλητές εξόδου την συγκέντρωση του αντιδρώντος A, C_A και την θερμοκρασία του αντιδραστήρα T. Στόχος του ρυθμιστή δηλαδή είναι να βρει το χρονικό προφίλ της ροής εισόδου και της θερμοκρασίας του ψυκτικού μέσου ώστε το σύστημα μας να φτάσει στην επιθυμητή κατάσταση $(C_{A,set}, T_{set})$. Στην περίπτωση αυτή το ψυκτικό μέσο είναι απαραίτητο γιατί η αντίδραση που λαμβάνει χώρα είναι ισχυρά εξώθερμη οπότε η θερμοκρασία του αντιδραστήρα μπορεί να ανέβει σε πολύ υψηλά επίπεδα χωρίς την βοήθεια ψυκτικού. Οι τιμές των διαφόρων παραμέτρων του συστήματος δίνονται στον παρακάτω πίνακα.

Παράμετρος Διεργασίας	Αριθμητική Τιμή	Μονάδα Μέτρησης
V	100	[lt]
UA	20000	[J/min K]
ρ	1000	[g/lt]
c_p	4.2	[J/g K]
$-(\Delta H_R)$	596619	[J/mol]
k_0	6.85E+11	[lt/min mol]
E	76534.704	[J/mol]
R	8.314	[J/mol K]
T_{in}	275	[K]
$C_{A,in}$	1	[mol/lt]

Πίνακας 6: Πίνακας παραμέτρων της διεργασίας

Θα πρέπει τώρα και πάλι να μετφράσουμε τα δεδομένα μας σε γλώσσα ενισχυτικής μάθησης. Ως κατάσταση του συστήματος θεωρούμε το τετράνυσμα:

$$\mathbf{s}(t) = (C_A(t), C_{A_{set}}, T(t), T_{set})$$

Ο λόγος που η κατάσταση ορίζεται με αυτόν τον τρόπο είναι παρόμοιος με το σύστημα μίας εξόδου και εισόδου που μελετήσαμε. Ο ρυθμιστής προκειμένου να λάβει μία δράση θα πρέπει να ξέρει τόσο την συγκέντρωση του A όσο και την θερμοκρασία του αντιδραστήρα αλλά και τις επιθυμητές προδιαγραφές μας, δηλαδή τα set points. Στο συγκεκριμένο πρόβλημα προσδιορίζουμε την κατάσταση του συστήματος (δηλαδή υπολογίζουμε τις τιμές

$C_A(t)$ και $T(t)$ σε κάθε χρονικό βήμα ενός επισοδείου λύνοντας το παραπάνω σύστημα μη-γραμμικών διαφορικών εξισώσεων με την ρουτίνα odeint της βιβλιοθήκης scipy της Python

Η συνάρτηση πολιτικής μας τώρα είναι η διανυσματική συνάρτηση:

$$\boldsymbol{\mu}(t) = (F(t), T_j(t))$$

και ομοίως με πριν ορίζουμε ως δράση την διανυσματική συνάρτηση:

$$\boldsymbol{\alpha}(t) = \boldsymbol{\mu}(t) + \mathcal{N}(t) = (F(t) + \mathcal{N}_0(t), T_j(t) + \mathcal{N}_1(t))$$

όπου και πάλι $\mathcal{N}(t) = (\mathcal{N}_1(t), \mathcal{N}_2(t))$ ο θόρυβος κατά Ornstein-Uhlenbeck. Εφόσον στο πρόβλημά μας τώρα ο χώρος των δράσεων είναι δύο διαστάσεων χρειαζόμαστε δύο τυχαίους περιπάτους προς εξερεύνηση του χώρου, έναν για κάθε δράση. Ομοίως με πριν, η δράση κάνει τον πράκτορα να μεταβεί από την τρέχουσα κατάσταση στην επόμενη, την οποία θα συμβολίσουμε ως:

$$\mathbf{s}'(t) = (C_A(t+1), C_{A_{set}}, T(t+1), T_{set})$$

Η συνάρτηση κέρδους, $r(s, \alpha, s')$, ορίζεται τώρα ως:

$$r(s,\alpha,s') = \begin{cases} 1 & ,|C_A(t+1) - C_{A_{set}}| < |C_A(t) - C_{A_{set}}| \kappa\alpha\iota |T(t+1) - T_{set}| < |T(t) - T_{set}| \\ 200 & ,\left|C_{A_{t+1}} - C_{A_{set}}\right| < 0.003 \quad \kappa\alpha\iota \quad |T(t+1), T_{set}| < 1 \\ -100 & ,T_j(t) + \mathcal{N}_1(t) > 300 \\ -10, \alpha\lambda\lambda\iota\omega\varsigma & \end{cases}$$

Η λογική της παραπάνω συνάρτησης κέρδους είναι η εξής: Δίνουμε στον πράκτορα μία μικρή αμοιβή (+1) κάθε φορά που τόσο η συγκέντρωση του A όσο και η θερμοκρασία του αντιδραστήρα πλησιάζουν κατ'απόλυτη τιμή τα επιθυμητά set points. Δίνουμε μία πολύ μεγάλη θετική αμοιβή (+200) όταν το σύστημά μας είναι ικανοποιητικά κοντά στα set points, έτσι ώστε να ωθήσουμε τον πράκτορα να παραμείνει κοντά τους για μεγαλύτερο χρονικό διάστημα. Επίσης δίνουμε στον πράκτορα μεγάλη αρνητική αμοιβή (-100) κάθε φορά που η δράση της θερμοκρασίας του ψυκτικού είναι πάνω από 300K. Αυτό γίνεται έτσι ώστε να αποφευχθούν για όσο το δυνατόν πιο πολλά χρονικά βήματα θερμοκρασίες ψυκτικού που είναι πάνω από 300K. Προσπαθούμε δηλαδή να περιορίσουμε την θερμοκρασία του ψυκτικού σε όσο στενότερα όρια γίνεται μέσω της συνάρτησης κέρδους. Τέλος αν καμία από αυτές τις συνθήκες δεν πληρείται δίνουμε στον πράκτορα μία αρνητική αμοιβή (-10). Ο λόγος που δίνουμε αρνητική αμοιβή μεγαλύτερη κατ'απόλυτη τιμή από +1 (η αμοιβή του πρώτου κλάδου της συνάρτησης) είναι για να ωθήσουμε τον πράκτορα να πλησιάσει πιο γρήγορα και πιο αποδοτικά τα set points, εφόσον ο στόχος του είναι να μαζέψει κατά το δυνατόν μεγαλύτερο κέρδος στο τέλος ενός επισοδείου.

Τώρα θα παρουσιάσουμε τις παραμέτρους της μεθοδολογίας που χρησιμοποιήθηκαν για την προσομοίωση:

Τα νευρωνικά δίκτυα του δράστη, του κριτή αλλά και τα αντίστοιχα δίκτυα-στόχοι έχουν το καθένα από δύο κρυφά στρώματα. Το πρώτο κρυφό στρώμα έχει 450 νευρώνες και το

δεύτερο έχει 400 νευρώνες. Όλες οι συναρτήσεις ενεργοποίησης επιλέχθηκαν και εδώ ως τύπου ReLU. Ο ρυθμός εκμάθησης για όλα τα δίκτυα είναι ίσος με $5 \cdot 10^{-4}$ και, ομοίως με πρίν $\tau = 10^{-3}$. Αυτήν την φορά δεν έχουμε επιβάλλει κάποιου είδους κανονικοποίησης κόστους. Ο παράγοντας προεξόφλησης είναι ίσος με 0.99. Χωρητικότητα μνήμης αναπαραγωγής ίση με 10^4, ίση και με τον χρόνο προθέρμανσης. Το μέγεθος των παρτίδων τώρα είναι ίσο με 128. Τα κρυφά στρώματα είναι κανονικοποιημένα. Χρησιμοποιήθηκε και πάλι το βήμα **17** της αντιστροφής βαθμίδας και τα όρια των δράσεων είναι τα εξής: $16 \leq F \leq 24$ και $200 \leq T_j \leq 300$. Σαν βελτιστοποιητής χρησιμοποιήθηκε και πάλι ο GradientDescentOptimizer. Στο σημείο αυτό κρίνεται σκόπιμο να αναφερθεί ότι στις αρχικές προσομοιώσεις του εν λόγω συστήματος είχε χρησιμοποιηθεί ο Adam Optimizer [30] του Tensorflow. Παρόλα αυτά ακόμη και μετά από αρκετές ώρες μέσα στην προσομοίωσης ο αλγόριθμος δεν κατάφερε να συγκλίνει. Για τον σκοπό αυτό δοκιμάσθηκε ο GradientDescentOptimizer και τα αποτελέσματα που θα παρουσιασθούν λήφθηκαν μετά από περίπου μίαμιση ώρα μέσα στην προσομοίωση.

Και εδώ ένα επισόδειο διαρκεί για 300 χρονικά βήματα. Ομοίως με την προηγούμενη περίπτωση ένα επισόδειο τελειώνει όταν:

- Το επισόδειο έχει ολοκληρωθεί, δηλαδή ο αλγόριθμος έχει τρέξει για 300 χρονικά βήματα
- Ο ρυθμιστής έχει καταφέρει να εντοπίσει το set point και έχει καταφέρει να παραμείνει ικανοποιητικά κοντά σε αυτό για 5 χρονικά βήματα, δηλαδή ικανοποιούνται οι ανισώσεις $\left|C_{A_{t+1}} - C_{A_{set}}\right| < 0.003$ και $|T(t+1), T_{set}| < 1$ για πέντε χρονικά βήματα

Και στην περίπτωση αυτή στην αρχή του κάθε επισοδείου θα πρέπει να επαναφέρουμε το σύστημα στην αρχική του κατάσταση , δηλαδή σε συγκέντρωση A ίση με την συγκέντρωση εισόδου και θερμοκρασία ίση με την θερμοκρασία εισόδου.

Στις παραπάνω παραμέτρους δεν έχουν συμπεριληφθεί οι παράμετροι του θορύβου Ornstein-Uhlenbeck, δηλαδή του τυχαίου περιπάτου. Ο λόγος είναι επειδή στο συγκεκριμένο σύστημα έχουμε εξετάσει πώς ο θόρυβος αυτός επηρεάζει την προσομοίωση και την συμπεριφορά του συστήματος. Έχουν γίνει δηλαδή δύο προσομοιώσεις -για το ίδιο set point- μία χωρίς θόρυβο και μία με θόρυβο, οι παράμετροι των οποίων θα αναφερθούν παρακάτω.

Το διάνυσμα του set point που θα θέσουμε για τις προσομοιώσεις που θα ακολουθήσουν είναι το εξής

$$s_{set} = (C_{A_{set}}, T_{set}) = (0.07, 375)$$

Τα set points αυτά δεν επιλέχθηκαν τυχαία. Συγκεκριμένα, το σημείο $s_{eq} = (C_{A_{eq}}, T_{eq}) = (0.0754, 376.3)$ αποτελεί *ευσταθές σημείο ισορροπίας* του δυναμικού συστήματος (9.2.1). Πράγματι αν στο σύστημα αυτό μηδενίσουμε τις παραγώγους, αντικαταστήσουμε τις παραμέτρους του πίνακα 6 αλλά και τις εξής τιμές της ροής εισόδου και της θερμοκρασίας του ψυκτικού

$$F_{eq} = 20 \frac{lt}{min}$$
$$T_{j,eq} = 250K$$

και λύσουμε το 2×2 σύστημα μη-γραμμικών εξισώσεων που προκύπτει τότε θα πάρουμε ως λύση το σημείο s_{eq}. Ο τρόπος με τον οποίο αποφανθήκαμε ότι το παραπάνω σημείο ισορροπίας είναι ευσταθές ήταν με την μέθοδο της *απειροστής μεταβολής των παραμέτρων*. Η μέθοδος αυτή συνίσταται στην εισαγωγή μίας μικρής διαταραχής στις παραπάνω μεταβλητές ισορροπίας F_{eq} και $T_{j,eq}$

$$F_{eq,pert} = F_{eq} + dF$$
$$T_{j,eq,pert} = T_{j,eq} + dT_j$$

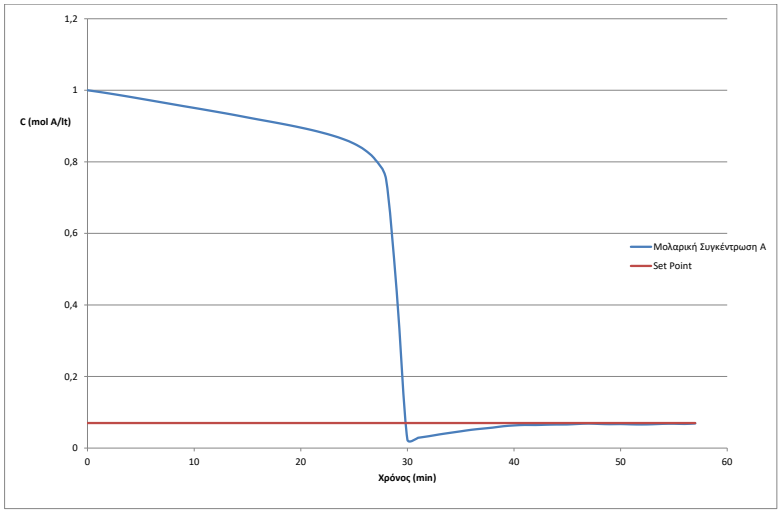
όπου οι μικρές αυτές διαταραχές είναι της τάξεως του 10^{-2}. Αν επιβάλλοντας αυτές τις διαταραχές παρατηρήσουμε ότι το σύστημα δίνει μία τελείως διαφορετική λύση, πολύ μακριά από το σημείο ισορροπίας, τότε το σημείο ισορροπίας χαρακτηρίζεται ως *ασταθές*. Αν αντιθέτως η λύση είναι κοντά στο σημείο ισορροπίας τότε χαρακτηρίζεται ως *ευσταθές*. Με την παραπάνω μέθοδο συμπεραίνουμε ότι το εν λόγω σημείο ισορροπίας s_{eq} είναι ευσταθές.

Συνεπώς το επιθυμητό set point της προσομοίωσης βρίσκεται σε μία γειτονιά του ευσταθούς σημείου ισορροπίας.

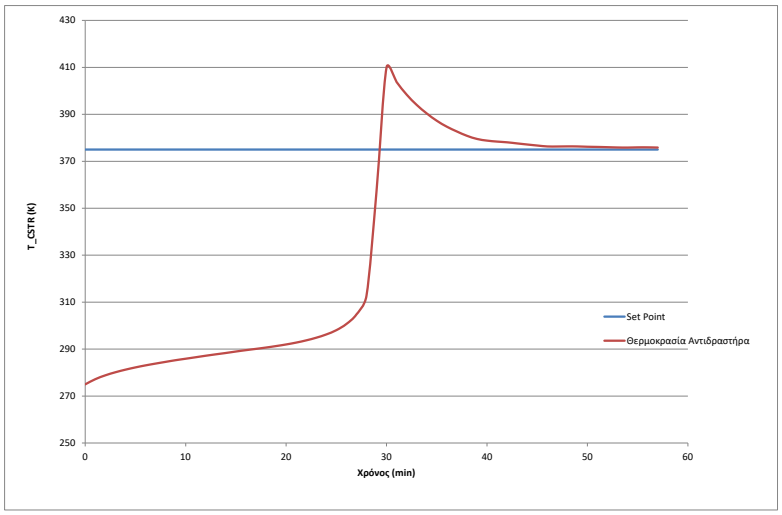
Ξεκινάμε τώρα με την παρουσίαση των αποτελεσμάτων:

- $C_{A_{set}} = 0.07$, $T_{set} = 375$ με την χρήση θορύβου κατά Ornstein-Uhlenbeck

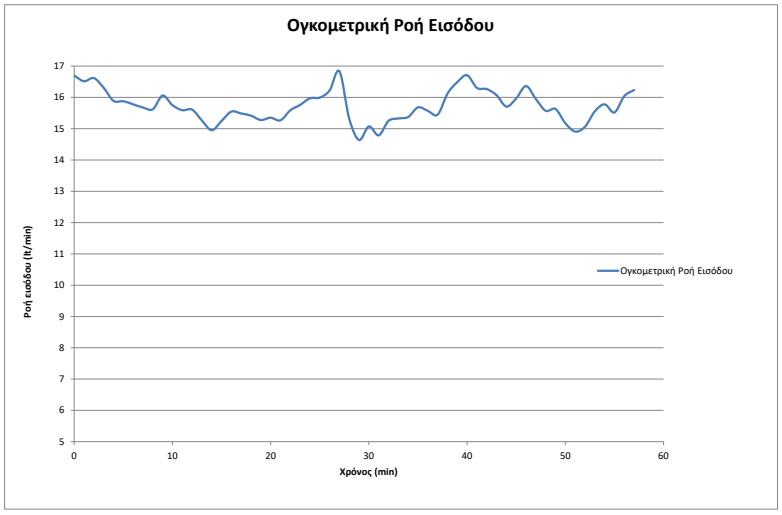
Οι παράμετροι του θορύβου που χρησιμοποιήθηκαν είναι $\mu = 0, \theta = 0.15, \sigma = 0.2$. Τα αποτελέσματα φαίνονται στα παρακάτω διαγράμματα:



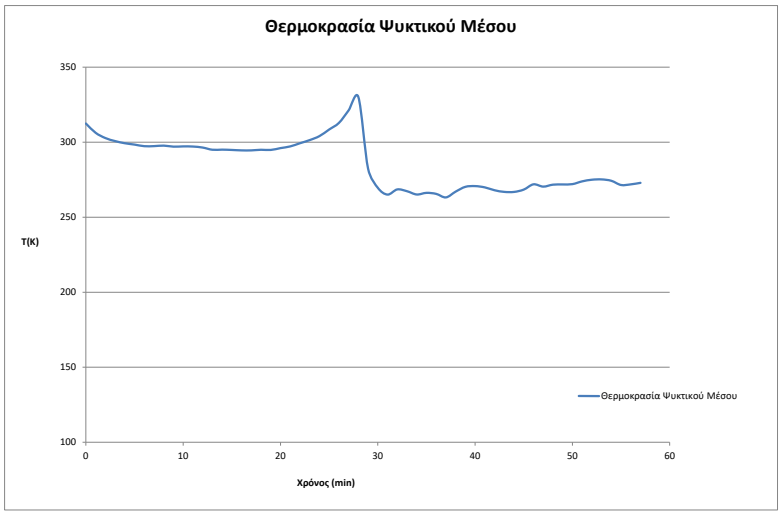
Εικόνα 14: Προφίλ της συγκέντρωσης του Α συναρτήσει του χρόνου με την χρήση θορύβου



Εικόνα 15: Προφίλ της θερμοκρασίας του αντιδραστήρα συναρτήσει του χρόνου με την χρήση θορύβου



Εικόνα 16: Προφίλ της ροής εισόδου συναρτήσει του χρόνου με την χρήση θορύβου



Εικόνα 17: Προφίλ της θερμοκρασίας του ψυκτικού μέσου συναρτήσει του χρόνου με την χρήση θορύβου

Ας παρατηρήσουμε για αρχή το τελικό αποτέλεσμα της εκπαίδευσης, δηλαδή τα διαγράμματα της συγκέντρωσης και της θερμοκρασίας του αντιδραστήρα. Βλέπουμε ότι περίπου μέχρι και το χρονικό βήμα 28 η συγκέντρωση του A πέφτει με ομαλό και αργό ρυθμό και συνεπώς, εφόσον η αντίδραση είναι εξώθερμη, η θερμοκρασία του αντιδραστήρα ανεβαίνει και αυτή με έναν ομαλό ρυθμό. Στα χρονικά βήματα 29 και 30 όμως παρατηρείται μία πολύ απότομη κλίση στο διάγραμμα της συγκέντρωσης, όπου και φαίνεται ότι ο ρυθμός της αντίδρασης είναι μέγιστος. Στα ίδια χρονικά βήματα βλέπουμε ότι και η θερμοκρασία παρουσιάζει μία απότομη κλίση, ως αποτέλεσμα της απότομης μείωσης της συγκέντρωσης. Μάλιστα παρατηρείται και μία υπέρβαση τόσο στην συγκέντρωση όσο και στην θερμοκρασία. Μετά από την υπέρβαση αυτή η συγκέντρωση ανεβαίνει και πάλι, προσεγγίζει το επιθυμητό set point με πολύ μεγάλη ακρίβεια και παραμένει πολύ κοντά σε αυτό (εφόσον η αντίδραση είναι μονόδρομη δεν μπορεί να παράγεται A μέσω της αντίδρασης οπότε το γιατί έχουμε αύξηση της συγκέντρωσής του A θα το εξηγήσουμε με βάση τα διαγράμματα των δράσεων). Μάλιστα φαίνεται ότι το επισόδειο τελειώνει λίγο πριν το χρονικό βήμα 60, που σημαίνει ότι πληρείται η δεύτερη προϋπόθεση τερματισμού ενός επισοδείου. Ίδιες παρατηρήσεις κάνουμε και για την θερμοκρασία του αντιδραστήρα. Μετά την υπέρβαση, η οποία φαίνεται να φτάνει μέχρι τους 410K, η θερμοκρασία πέφτει, προσεγγίζει το set point και παραμένει πολύ κοντά σε αυτό. Συνεπώς φαίνεται ότι έχει επιτευχθεί μόνιμη κατάσταση, όπως και επιθυμούσαμε. Η μόνιμη αυτή κατάσταση μας εγγυάται μετατροπή του A ίση με 93% και θερμοκρασία εξόδου του αντιδραστήρα ίση με 375K.

Συνεχίζουμε με τα διαγράμματα των δράσεων. Παρατηρούμε ότι ο ρυθμιστής μας δίνει

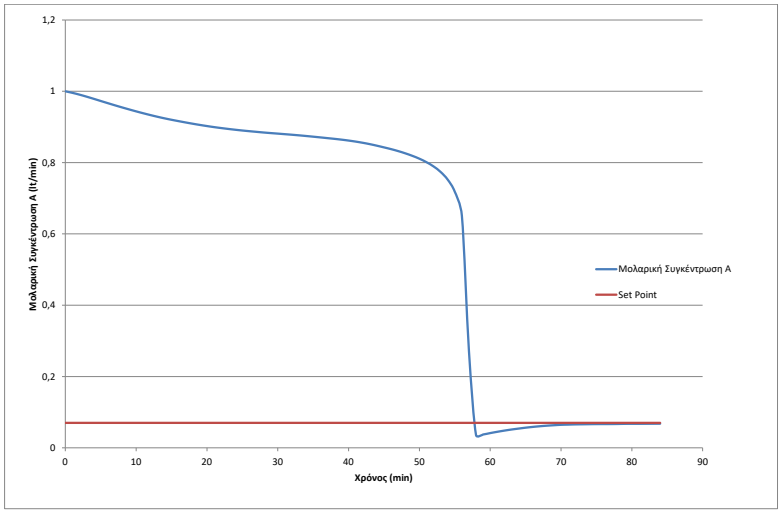
ένα αρκετά θορυβώδες προφίλ για την ροή εισόδου στον αντιδραστήρα. Το προφίλ αυτό δεν φαίνεται να παρουσιάζει κάποιο μοτίβο. Η μόνη παρατήρηση που μπορούμε να κάνουμε είναι ότι στα ίδια χρονικά βήματα που παρατηρείται η απότομη κλίση στο διάγραμμα της συγκέντρωσης και της θερμοκρασίας ο ρυθμιστής δίνει την εντολή για μία αυξημένη ροή εισόδου. Σε αυτήν την αυξημένη ροή οφείλεται και η αύξηση της συγκέντρωσης του A αμέσως μετά την υπέρβαση. Εφόσον η αντίδραση είναι μονόδρομη, δεν μπορεί να παράγεται A από την αντίδραση. Παρόλα αυτά η αυξημένη ροή εισόδου έχει ως αποτέλεσμα να εισέρχεται στον αντιδραστήρα μεγαλύτερη ποσότητα A από αυτήν που παράγεται και εξέρχεται. Μέτα από το σημείο αυτό πάλι φαίνεται πως η ροή εισόδου δεν παρουσιάζει κάποιο συγκεκριμένο μοτίβο. Όσον αφορά την δράση της θερμοκρασίας του ψυκτικού φαίνεται ότι στα αρχικά στάδια του επισοδείου παραμένει σχεδόν σταθερή (αρχίζοντας από τους 310K και στην συνέχεια πέφτοντας γύρω στους 300K). Και σε αυτήν την περίπτωση στα χρονικά βήματα της απότομης κλίσης της συγκέντρωσης και της θερμοκασίας ο ρυθμιστής μας δίνει την εντολή για μία απότομη αλλαγή στην θερμοκρασία του ψυκτικού, η οποία παρουσιάζει ιδιαίτερο ενδιαφέρον. Σκοπός του ψυκτικού είναι να ρυθμίσει τις αυξήσεις της θερμοκρασίας του αντιδραστήρα λόγω του εξώθερμου της αντίδρασης. Παρόλα αυτά φαίνεται ότι εδώ ο ρυθμιστής δίνει εντολή για αύξηση θερμοκρασίας του ψυκτικού, παρά της αύξηση της θερμοκρασίας λόγω της πορείας της αντίδρασης. Σε αυτήν την δράση οφείλεται και η υπέρβαση στην θερμοκρασία του αντιδραστήρα και στην συγκέντρωση του A. Φυσικά πρέπει να θυμόμαστε ότι ο ρυθμιστής δεν έχει καμία ενσωματωμένη έννοια του τί σημαίνει το να είναι μία αντίδραση εξώθερμη, ενδόθερμη ή της φυσικής σημασίας του συστήματος. Ο μοναδικός στόχος του ρυθμιστή είναι να κάνει ό,τι είναι απαραίτητο για να φτάσει το σύστημα μας στο επιθυμητό set point. Το μόνο που μπορούμε να κάνουμε είναι να τον καθοδηγήσουμε όσο πιο επιτυχημένα μπρορούμε στα set points αυτά με κατάλληλες συναρτήσεις κέρδους και άλλες παραμέτρους της μεθόδου. Μετά από την αύξηση αυτή της θερμοκρασίας του ψυκτικού ο ρυθμιστής δίνει την εντολή για απότομη μείωση της θερμοκρασίας, μιας και ανιχνεύει ήδη μία πολύ μεγάλη (σε σχέση με την επιθυμητή) θερμοκρασία του αντιδραστήρα. Μετά από την μείωση αυτή η θερμοκρασία του ψυκτικού φαίνεται να σταθεροποιείται σε μία θερμοκρασία κοντά στην περιοχή των 275K, όπου και επιτυγχάνεται η μόνιμη κατάσταση.

Το συμπέρασμα που εξάγουμε είναι ότι σε καθαρά θεωρητικό επίπεδο η ρύθμιση του συστήματος με ρυθμιστή βασισμένο στην προτεινόμενη μεθοδολογία είναι επιτυχής αφού η μόνιμη κατάσταση των μεταβλητών εξόδου επιτυγχάνεται με πολύ μεγάλη ακρίβεια. Παρόλα αυτά κρίνεται σκόπιμο να αναφερθεί ότι σε πρακτικό επίπεδο τα πράγματα ίσως να μην είναι τόσο εύκολα. Για αρχή θα πρέπει να είμαστε σίγουροι ότι ο αντιδραστήρας μας θα μπορεί να αντέξει αυτές τις υπερβάσεις της θερμοκρασίας. Επίσης θα πρέπει η βιομηχανία να μπορεί να διαθέσει ψυκτικό που να φτάνει τις θερμοκρασίες που δίνει ο ρυθμιστής. Τέλος θα πρέπει να ελεγθεί κατά πόσο είναι εφικτό αυτό το θορυβώδες προφίλ της ροής εισόδου στον αντιδραστήρα να επιτευχθεί στην πράξη.

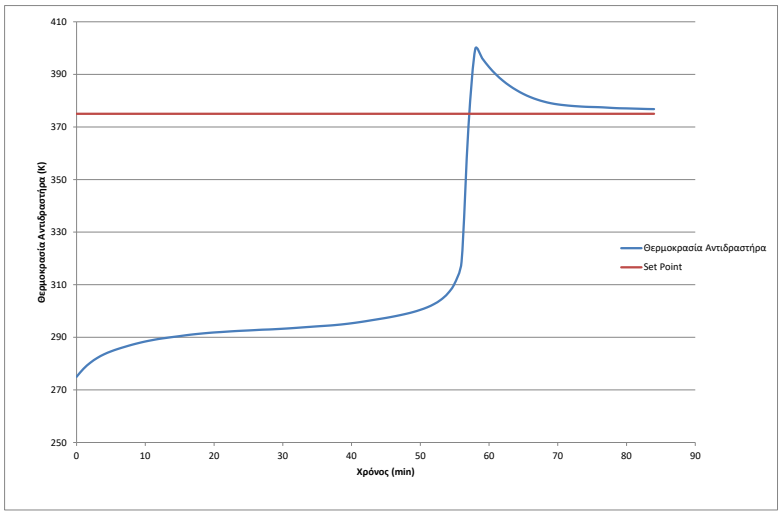
Συνεχίζουμε με τα αποτελέσματα που προκύπτουν χωρίς την εισαγωγή του θορύβου κατά Ornstein-Uhlenbeck:

- $C_{A_{set}} = 0.07$, $T_{set} = 375$ χωρίς την χρήση θορύβου κατά Ornstein-Uhlenbeck

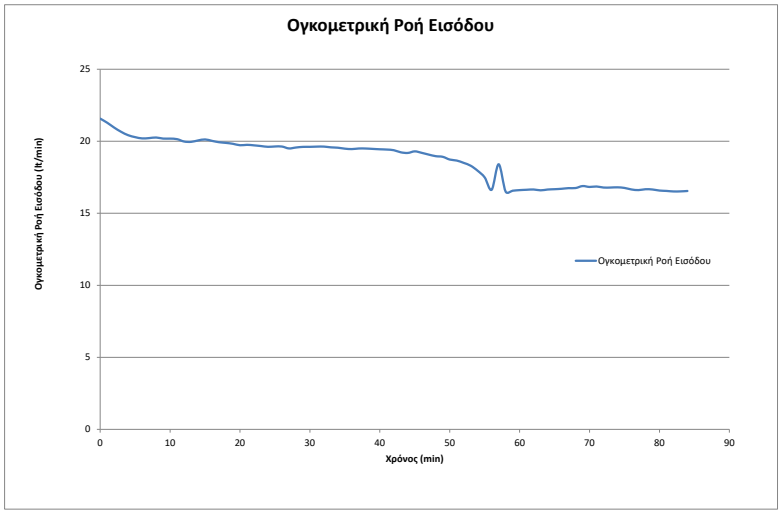
Όλες οι υπόλοιπες παράμετροι της προσομοίωσης παραμένουν οι ίδιες. Η μόνη αλλαγή που κάνουμε είναι να αφαιρέσουμε την συνιστώσα του θορύβου Ornstein-Uhlenbeck από την δράση. Τα καινούρια αποτελέσματα φαίνονται στα παρακάτω διαγράμματα:



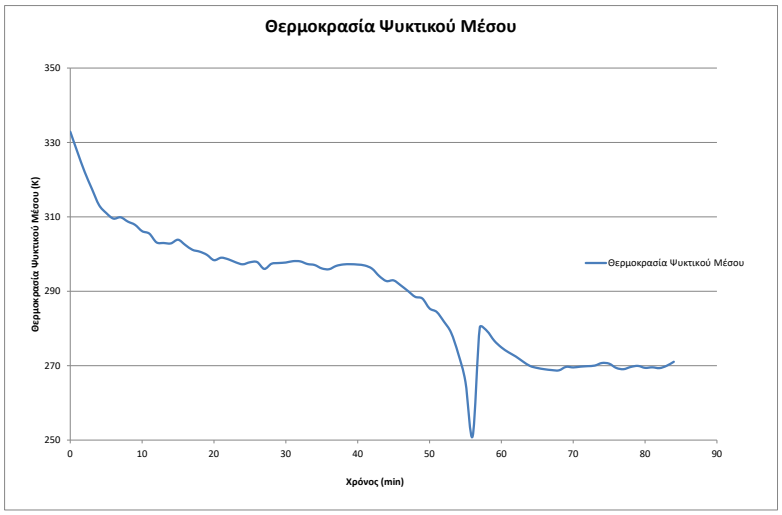
Εικόνα 18: Προφίλ της συγκέντρωσης του Α συναρτήσει του χρόνου χωρίς την χρήση θορύβου



Εικόνα 19: Προφίλ της θερμοκρασίας του αντιδραστήρα συναρτήσει του χρόνου χωρίς την χρήση θορύβου



Εικόνα 20: Προφίλ της ροής εισόδου συναρτήσει του χρόνου χωρίς την χρήση θορύβου



Εικόνα 21: Προφίλ της θερμοκρασίας του ψυκτικού συναρτήσει του χρόνου χωρίς την χρήση θορύβου

Με μία γρήγορη ματιά στα παραπάνω διαγράμματα τα αποτελέσματα είναι πολύ πιο αισιόδοξα, κυρίως όσον αφορά το προφίλ των δράσεων, σε σχέση με αυτά που προέκυψαν με την εισαγωγή της συνιστώσας του θορύβου Ornstein-Uhlenbeck. Ας αρχίσουμε όμως για αρχή να παρατηρούμε τα διαγράμματα των μεταβλητών εξόδου.

Παρατηρούμε ότι η συμπεριφορά τόσο της συγκέντρωσης του A όσο και της θερμοκρασίας του αντιδραστήρα είναι πολύ παρόμοια με αυτήν που προέκυψε από την χρήση του θορύβου. Υπάρχει όμως μία ουσιώδης διαφορά. Η περιοχή της απότομης μεταβολής της συγκέντρωσης και της θερμοκρασίας βρίσκεται τώρα κοντά στο χρονικό βήμα 58 και 59, σε αντίθεση με την προηγούμενη περίπτωση όπου παρατηρούσαμε αυτήν την απότομη αλλαγή περίπου στο χρονικό βήμα 29. Αυτή η χρονική καθυστέρηση έχει μία θετική συνέπεια στην εκπαίδευση και την ρύθμιση του συστήματος. Αν συγκρίνουμε, για παράδειγμα, τα διαγράμματα της συγκέντρωσης του A με και χωρίς την χρήση θορύβου θα παρατηρήσουμε ότι στο σημείο που αρχίζει η απότομη κλίση του διαγράμματος στην περίπτωση της μη-χρήσης του θορύβου είναι μικρότερη σε σχέση με την συγκέντρωση στην περίπτωση του θορύβου. Αυτό συμβαίνει γιατί, λόγω της χρονικής αυτής υστέρησης στην δεύτερη περίπτωση, η αντίδραση προλαβαίνει να προχωρήσει λίγο παραπάνω, ρίχνοντας έτσι περισσότερο την συγκέντρωση του A. Η θετική συνέπεια αυτής της παρατήρησης είναι το γεγονός ότι η υπέρβαση που παρατηρείται, τόσο στην συγκέντρωση όσο και στην θερμοκρασία, είναι μικρότερη σε σχέση με την υπέρβαση που παρατηρούσαμε όταν ήταν παρών και ο θόρυβος. Στην περίπτωση της θερμοκρασίας του αντιδραστήρα, για παράδειγμα, ο θόρυβος οδηγούσε σε μία υπέρβαση που έφτανε τους 410K ενώ η εξαγωγή του θορύβου μείωσε την υπέρβαση αυτή κατά 10

περίπου βαθμούς. Φυσικά και σε αυτήν την περίπτωση θα πρέπει να σιγουρευτούμε ότι ο αντιδραστήρας μας θα μπορεί να αντέξει τις υψηλές αυτές θερμοκρασίες. Μετά από αυτές τις υπερβάσεις, η συγκέντρωση και η θερμοκρασία επανέρχονται με ομαλό τρόπο στα set points και συγκλίνουν σε αυτά. Συνεπώς και σε αυτήν την περίπτωση πετύχαμε μόνιμη κατάσταση και η ρύθμιση ήταν επιτυχής, σε θεωρητικό επίπεδο προς το παρόν. Μέχρι τώρα εξετάσαμε το αποτέλεσμα του ρυθμιστή μας. Συνεχίζουμε τώρα με τα διαγράμματα των δράσεων, δηλαδή τις εντολές του ίδιου του ρυθμιστή.

Για αρχή δεν μπορεί κανείς παρά να παρατηρήσει την τεράστια διαφορά του προφίλ της ροής εισόδου του αντιδραστήρα. Στην τρέχουσα περίπτωση η ροή εισόδου παρουσιάζει μία πολύ πιο ομαλή συμπεριφορά η οποία έχει πάψει να χαρακτηρίζεται από το θορυβώδες προφίλ της περίπτωσης του θορύβου. Στην αρχή του επισοδείου, όπου παρατηρείται και η ομαλή μεταβολή της συγκέντρωσης, η ροή εισόδου φαίνεται να έχει μία πτωτική -σχεδόν σταθερή- συμπεριφορά. Η μόνη παρατήρηση που μπορούμε να κάνουμε είναι η εντολή της πτώσης της ροής και της μικρής κορυφής που δίνει ο ρυθμιστής ακριβώς στην περιοχή της απότομης μεταβολής των μεταβλητών εξόδου. Είναι και ο λόγος που μετά την μικρή υπέρβαση η συγκέντρωση του A ανεβαίνει. Μετά την κορυφή αυτή η ροή εισόδου φτάνει και αυτή σε μόνιμη κατάσταση και σταθεροποιείται περίπου στα $16.5\frac{lt}{min}$.

Όσον αφορά την θερμοκρασία του ψυκτικού φαίνεται να παρουσιάζει σχεδόν εξ'ολοκλήρου μία πτωτική συμπεριφορά. Όταν η συγκέντρωση του A μεταβάλλεται πολύ απότομα (κοντά στο χρονικό βήμα 58) ο ρυθμιστής δίνει την εντολή για μία ραγδαία πτώση της θερμοκρασία του ψυκτικού. Η συμπεριφορά αυτή είναι πιο κοντά στην διαίσθησή μας. Εφόσον η αντίδραση είναι εξώθερμη, στις περιοχές απότομης μείωσης της συγκέντρωσης (δηλαδή σε περιοχές μεγάλου ρυθμού αντίδρασης) αναμένουμε και μεγάλες αυξήσεις στην θερμοκρασία (όπως επιβεβαιώνεται και από το διάγραμμα της θερμοκρασίας του αντιδραστήρα). Συνεπώς εκείνο είναι το σημείο όπου ο ρυθμιστής πρέπει να λάβει δράση για να μειώσει την θερμοκρασία του αντιδραστήρα όσο πιο γρήγορα μπορεί. Εμφανίζεται λοιπόν αυτή η μικρή κορυφή στην θερμοκρασία του ψυκτικού, η οποία διαρκεί για μόλις δύο χρονικά βήματα. Αυτή είναι και η μεγαλύτερη διαφορά που μπορούμε να διακρίνουμε όταν συγκρίνουμε τα διαγράμματα της συμπεριφοράς της θερμοκρασίας του ψυκτικού. Στην περίπτωση του θορύβου ο ρυθμιστής δίνει εντολή για μία *αύξηση* της θερμοκρασίας του ψυκτικού, κάτι που όντως προκαλεί έκπληξη στην αρχή. Αυτή η αύξηση της θερμοκρασίας είναι και ο λόγος που η υπέρβαση στην περίπτωση του θορύβου είναι μεγαλύτερη από την υπέρβαση της θερμοκρασίας όταν δεν έχουμε θόρυβο, όπου ο ρυθμιστής δίνει εντολή για μείωση της θερμοκρασίας του ψυκτικού στην περιοχή υψηλού ρυθμού αντίδρασης. Μετά από την κορυφή αυτή, ο ρυθμιστής αυξάνει και πάλι την θερμοκρασία του ψυκτικού και στην συνέχεια την ρίχνει με ομαλό ρυθμό, όπου και συγκλινει κοντά στους 270K.

Αυτό που μπορούμε να συμπεράνουμε είναι ότι η εξαγωγή της συνιστώσας του θορύβου κατά Ornstein-Uhlenbeck έχει αρκετά πλεονεκτήματα και ένα μειονέκτημα. Το βασικό μειονέκτημα είναι ότι η ρύθμιση του συστήματος παρουσιάζει μία χρονική υστέρηση της τάξεως των 20 χρονικών βημάτων, που στην συγκεκριμένη περίπτωση σημαίνει χρονική

υστέρηση 20 λεπτών. Τα 20 λεπτά αυτά μπορεί να είναι κρίσιμα για την διεργασία ή/και για το χρονοδιάγραμμα της βιομηχανίας που χρησιμοποιεί τον προτεινόμενο ρυθμιστή. Στα πλεονεκτήματα φυσικά ανήκουν οι μικρότερες υπερβάσεις των μεταβλητών εξόδου καθώς και τα πολύ πιο ομαλά προφίλ των μεταβλητών εισόδου (ροή και θερμοκρασία ψυκτικού). Το δεύτερο είναι αρκετά πιο σημαντικό καθώς τώρα μπορεί να μπει και στο παιχνίδι η πιθανή πρακτική εφαρμογή ενός τέτοιου ρυθμιστή.

Είναι δύσκολο να εξηγηθεί με πλήρη μαθηματική ακρίβεια *γιατί* τα αποτελέσματα που παίρνουμε όταν βγάζουμε τον θόρυβο κατά Ornstein-Uhlenbeck είναι διαφορετικά (και στην συγκεκριμένη περίπτωση καλύτερα). Η ουσία είναι ότι με την χρήση του θορύβου ο πράκτορας-ρυθμιστής *εξερευνεί* τον χώρο με διαφορετικό τρόπο σε σχέση με την περίπτωση απουσίας του. Η διαφορετική αυτή εξερεύνηση ενδέχεται να δίνει στον πράκτορα έναν διαφορετικό τρόπο βελτιστοποίησης και ρύθμισης του συστήματος, ο οποίος να επιφέρει *μεγαλύτερο κέρδος στο τέλος του επισοδείου*. Θα πρέπει εν τέλει να θυμόμαστε ότι ο πράκτορας δεν έχει καμία έννοια του τι σημαίνει φυσικό σύστημα, φυσική σημασία κ.τ.λ. Πρόκειται για ένα πρόγραμμα στον υπολογιστή το οποίο είναι προγραμματισμένο να μαζέψει κατά το δυνατόν περισσότερο κέρδος ελαχιστοποιώντας τις συναρτήσεις κόστους. Κατά πάσα πιθανότητα ο πράκτορας μαζεύει περισσότερο κέρδος στην περίπτωση του θορύβου όταν εξερευνεί τον χώρο με τον τρόπο που είδαμε. Φυσικά ο τρόπος αυτός έχει ως αποτέλεσμα ένα χειρότερο σύστημα ρύθμισης, από πρακτικής μεριάς.

9 Συμπεράσματα και σχολιασμοί

Στην εργασία αυτή ο υβριδικός αλγόριθμος DDPG που συνδυάζει δύο πολύ σύγχρονες τεχνολογίες μηχανικής μάθησης, την μάθησης βάθους και την ενισχυτική μάθηση εφαρμόστηκε για την ρύθμιση συστημάτων χημικής μηχανικής και για πρώτη φορά σε πολυμεταβλητά συστήματα. Τα αποτελέσματα είναι ενθαρρυντικά, καθώς ο αλγόριθμος είναι ικανός με μηδενική αρχική γνώση της διεργασίας να μπορέσει ταυτόχρονα να εξερευνήσει τη δυναμική της και να την οδηγήσει σε επιθυμητά σημεία ισορροπίας. Παρακάτω παρουσιάζονται συγκεντρωτικά τα συμπεράσματα που προέκυψαν από την μελέτη που προηγήθηκε:

- Η συνάρτηση κέρδους $r(s, \alpha, s')$ είναι μία πολύ ευαίσθητη παράμετρος του αλγορίθμου και πολλές φορές μπορεί να χρειαστεί αρκετή σκέψη και δοκιμές μέχρι να βρεθεί η κατάλληλη. Δεν αποκλείεται να υπάρχουν συναρτήσεις κέρδους που να δίνουν καλύτερα αποτελέσματα σε σχέση με αυτά που παρουσιάσθηκαν σε αυτήν την εργασία

- Η εισαγωγή του θορύβου κατά Ornstein-Uhlenbeck δεν συνεπάγεται απαραίτητα καλύτερη εκπαίδευση με την πρακτική έννοια της ρύθμισης του συστήματος. Παρόλο που μπορεί να προσφέρει αποδοτικότερη και ταχύτερη εξερεύνηση του χώρου στον οποίο ζει η διεργασία δεν σημαίνει ότι απο πρακτικής σκοπιάς θα είναι και η προτιμότερη επιλογή

- Οι υπολογιστικοί χρόνοι είναι πολύ μικροί αλλά αυξάνονται σε πολυμεταβλητά και μη-γραμμικά συστήματα. Ενδεικτικά, στην εν λόγω εργασία, στο μονομεταβλητό 1×1 σύστημα ο αλγόριθμος μπόρεσε να συγκλίνει και να επιτύχει μόνιμη κατάσταση εντός περίπου μισής ώρας (ίσως και λίγο λιγότερο). Στο πολυμεταβλητό σύστημα ο αλγόριθμος χρειάστηκε πολύ περισσότερο χρόνο για να δώσει τα παραπάνω αποτελέσματα, περίπου μιάμιση ώρα. Παρατηρήθηκε επίσης ότι η εισαγωγή του θορύβου μείωσε για πολύ λίγο τον υπολογιστικό χρόνο

- Παρατηρούμε ότι στην περίπτωση του πολυμεταβλητού συστήματος το βήμα της αντιστροφής της βαθμίδας βελτιώνει την απόδοση του αλγορίθμου, αλλά δεν πετυχαίνει εξ'ολοκλήρου τον στόχο του. Για παράδειγμα η θερμοκρασία του ψυκτικού ξεπερνά σε κάθε προσομοίωση τους 300K (είτε στην αρχή του επισοδείου είτε πιο μετά), παρόλο που σαν άνω φράγμα έχουμε θέσει τους 300K. Ο λόγος που συμβαίνει αυτό είναι το γεγονός ότι ο ρυθμιστής έχει ανάγκη αυτές τις υψηλές θερμοκρασίες προκειμένου να οδηγήσει το σύστημα στην επιθυμητή κατάσταση ισορροπίας. Επειδή εμείς προσπαθούμε να περιορίσουμε κατά το δυνατόν περισσότερο τις δράσεις μας (για λόγους οικονομίας και ασφάλειας) δεν σημαίνει ότι η διεργασία είναι εφικτή στα πλαίσια που έχουμε ορίσει. Επίσης το σύστημά μας περιλαμβάνει πολλούς μη-γραμμικούς όρους με παραγώγους που δεν παρουσιάζουν πολύ καλή συμπεριφορά. Αυτός είναι ένας επιπλέον λόγος που δυσκολεύει ακόμα πιο πολύ τον ρόλο του βήματος αυτού. Παρόλα αυτά βλέπουμε από τα διαγράμματα ότι αυτές οι υπερβάσεις της θερμοκρασίας του ψυκτικού πάνω από τους 300K δεν συμβαίνουν παρά μόνο για ελάχιστα χρονικά βήματα, κάτι στο οποίο κατά πάσα πιθανότητα βοήθησε το βήμα της αντιστροφής βαθμίδας.

- Η κατάλληλη επιλογή βελτιστοποιητή είναι πολύ σημαντική για την απόδοση του αλγορίθμου. Για παράδειγμα ο Adam Optimizer έδειξε πολύ κακή συμπεριφορά στο πολυμεταβλητό σύστημα ενώ αντίθετα προέκυψαν πολύ καλά αποτελέσματα από την εφαρμογή του GradientDescentOptimizer. Το ποιος βελτιστοποιητής είναι ο καταλληλότερος είναι θέμα εμπειρίας και δοκιμής και σφάλματος

- Σε όλες τις υπο εξέταση περιπτώσεις παρατηρείται μία υπέρβαση πάνω ή κάτω από το επιθυμητό set point. Αυτή η παρατήρηση βέβαια ισχύει σε πολλές περιπτώσεις ρύθμισης κλειστού βρόγχου. Το επόμενο βήμα λοιπόν είναι η λεπτομερέστερη διερεύνηση των παραμέτρων του αλγορίθμου για να διαπιστωθεί ποια είναι αυτή που οδηγεί σε αυτές τις υπερβάσεις με σκοπό φυσικά την μείωση (ή ακόμα και την εξάλειψη) του μέτρου των υπερβάσεων αυτών. Μία πρώτη σκέψη είναι να ελέγξουμε ποια είναι η κοινή παράμετρος ή οι κοινές παράμετροι που δεν άλλαξαν κατά την διάρκεια των προσομοιώσεων και να δοκιμάσουμε να αλλάξουμε αυτές. Στην προκειμένη περίπτωση σε όλες τις προσομοιώσεις χρησιμοποιήθηκε η ίδια συνάρτηση κόστους (συνάρτηση μέσου τετραγωνικού σφάλματος), ο ίδιος βελτιστοποιητής (GradientDescentOptimizer) και ο ίδιος ρυθμός εκμάθησης (η παράμετρος τ) των νευρωνικών δικτύων. Συνεπώς

προτείνεται ο πειραματισμός και περαιτέρω προσομοιώσεις με την μεταβολή μίας κάθε φορά από τις παραπάνω παραμέτρους (και σε επόμενο στάδιο συνδυασμός αυτών ή και περισσότερων παραμέτρων).

- Η προσέγγιση που παρουσιάστηκε σε αυτήν την εργασία βρίσκεται ακόμη στα πρώτα πειραματικά στάδια και δεν είναι έτοιμη να εφαρμοστεί σε βιομηχανική κλίμακα. Είναι όμως μία πολύ υποσχόμενη προσέγγιση που βασίζεται σε σύγχρονους αλγορίθμους της επιστήμης των δεδομένων και σε γρήγορους επεξεργαστές τύπου GPU, η οποία θα εξελιχθεί περαιτέρω τα επόμενα χρόνια προκειμένου να αποτελέσει μία ασφαλή επιλογή για την ρύθμιση των διεργασιών ειδικά σε περιπτώσεις όπου οι συμβατικές τεχνικές μοντελοποίησης δεν μπορούν να προσφέρουν μοντέλα κατάλληλα για ενσωμάτωση σε αλγορίθμους αυτόματου ελέγχου ή όταν η δυναμική της διεργασίας μεταβάλλεται χρονικά.

10 Παράρτημα

10.1 Θεώρημα Σταθερού Σημείου του Banach

Για να διατυπώσουμε αλλά και να αποδείξουμε το Θεώρημα Σταθερού Σημείου του Banach θα πρέπει να δώσουμε κάποιους ορισμούς αλλά και να αποδείξουμε δύο μικρά λήμματα.

Ορισμός: Έστω $(\mathbf{X}, d)$ ένας μετρικός χώρος. Μία απεικόνιση $\mathbf{T} : \mathbf{X} \to \mathbf{X}$ καλείται *συστολή* στον $\mathbf{X}$ αν υπάρχει $c \in [0, 1)$ τέτοιος ώστε:

$$d(\mathbf{T}(x), \mathbf{T}(y)) \leq cd(x, y), \qquad \forall x, y \in \mathbf{X}$$

Λήμμα 1: Έστω $(\mathbf{X}, d)$ ένας μετρικός χώρος, $\mathbf{T}$ μία συστολή στον $\mathbf{X}$ και $x_0 \in \mathbf{X}$. Ορίζουμε ακολουθία $\{x_n\}$ στον $\mathbf{X}$ ως $x_n = \mathbf{T}(x_{n-1})$. Τότε ισχύει ότι

$$d(x_{n+1}, x_n) \leq c^n d(x_1, x_0) \qquad \forall n \in \mathbb{N} \tag{10.1.1}$$

Απόδειξη:
Η απόδειξη θα γίνει με επαγωγή στο n.

- $n = 0$

$$d(x_2, x_1) = d(\mathbf{T}(x_1), \mathbf{T}(x_2)) \leq c^1 d(x_1, x_0)$$

Έστω τώρα ότι ισχύει για τυχαίο $n \in \mathbb{N}$, δηλαδή έχουμε σαν δεδομένο ότι $d(x_{n+1}, x_n) \leq c^n d(x_1, x_0)$ (επαγωγική υπόθεση). Θα δείξουμε ότι ισχύει για $n + 1$. Πράγματι

$$d(x_{n+2}, x_{n+1}) = d(\mathbf{T}(x_{n+1}), \mathbf{T}(x_n)) \leq cd(x_{n+1}, x_n) \leq c^{n+1} d(x_1, x_1)$$

και η απόδειξη του λήμματος ολοκληρώθηκε.

Ορισμός (Ακολουθία Cauchy): Έστω $(\mathbf{X}, d)$ ένας μετρικός χώρος και $\{x_n\}$ ακολουθία ορισμένη στον $\mathbf{X}$. Θα λέμε ότι η $\{x_n\}$ είναι *ακολουθία Cauchy* αν $\forall \epsilon > 0$ $\exists n_0$ ώστε $\forall m > n > n_0$ να ισχύει $d(x_m, x_n) < \epsilon$.

Λήμμα 2: Έστω $(\mathbf{X}, d)$ ένας μετρικός χώρος, $\mathbf{T}$ μία συστολή στον $\mathbf{X}$ και $x_0 \in \mathbf{X}$. Η ακολουθία $\{x_n\}$ που ορίζεται ως $x_n = \mathbf{T}(x_{n-1})$ είναι ακολουθία Cauchy.
Απόδειξη:
Έστω $m, n, n_0 \in \mathbb{N}$ με $m > n > n_0$. Με την βοήθεια του προηγούμενου λήμματος και την τριγωνικής ανισότητας παίρνουμε

$$\begin{aligned} d(x_m, x_n) &\leq d(x_{m-1}, x_{m-1}) + d(x_{m-1}, x_{m-2}) + \cdots + d(x_{n+1}, x_n) \\ &\leq c^{m-1} d(x_1, x_0) + c^{m-2} d(x_1, x_0) + \cdots + c^n d(x_1, x_0) \\ &= c^n d(x_1, x_0) \sum k = 0^{m-n-1} c^k \leq c^n d(x_1, x_0) \sum_{k=0}^{\infty} c^k \\ &= c^n d(x_1, x_0) \left(\frac{1}{1-c} \right) \end{aligned}$$

Έστω τώρα τυχαίο $\epsilon > 0$. Επιλέγουμε τώρα $n_0 \in \mathbb{N}$ ώστε να ισχύει

$$c^{n_0} < \frac{\epsilon(1-c)}{d(x_1, x_0)}$$

Τέλος επιλέγουμε τα m, n να είναι μεγαλύτερα του n_0 και λαμβάνουμε

$$d(x_m, x_n) \leq c^n d(x_1, x_0) \left(\frac{1}{1-c}\right) < \frac{\epsilon(1-c)}{d(x_1, x_0)} d(x_1, x_0) \left(\frac{1}{1-c}\right) = \epsilon$$

Άρα πράγματι η ακολουθία $\{x_n\}$ είναι Cauchy και η απόδειξη ολοκληρώθηκε.

Ο λόγος που χρειαζόμαστε η παραπάνω ακολουθία να είναι Cauchy είναι επειδή το Θεώρημα Σταθερού Σημείου του Banach μιλάει για μετρικούς χώρους οι οποίοι είναι *πλήρεις*. Δίνουμε τον ορισμό:

Ορισμός: Ένας μετρικός χώρος $(\mathbf{X}, d)$ θα λέγεται *πλήρης* αν και μόνον αν κάθε ακολουθία Cauchy είναι συγκλίνουσα. Δηλαδή κάθε ακολουθία Cauchy είναι συγκλίνουσα και αντιστρόφως. Παραδείγματα πλήρων μετρικών χώρων είναι τα σύνολα $\mathbb{R}^n$ και $\mathbb{C}^n$ εφοδιασμένα με την μετρική της απόλυτης τιμής.

Έχουμε τώρα όλα όσα χρειαζόμαστε για να διατυπώσουμε και να αποδείξουμε το Θεώρημα Σταθερού Σημείου του Banach.

Θεώρημα Σταθερού Σημείου του Banach: Έστω $(\mathbf{X}, d)$ ένας μη-κενός πλήρης μετρικός χώρος και $\mathbf{T} : \mathbf{X} \to \mathbf{X}$ μία συστολή ορισμένη στον $\mathbf{X}$. Τότε η $\mathbf{T}$ έχει μοναδικό σταθερό σημείο, δηλαδή υπάρχει μοναδικό $x^* \in \mathbf{X}$ τέτοιο ώστε $\mathbf{T}(x^*) = x^*$. Μάλιστα ισχύει το εξής: επιλέγουμε τυχαία ένα σημείο $x_0 \in \mathbf{X}$ και ορίζουμε μία ακολουθία $\{x_n\}$ ως $x_n = \mathbf{T}(x_{n-1})$. Τότε ισχύει ότι $x_n \to x^*$.

Απόδειξη:
Εφόσον ο μετρικός χώρος $(\mathbf{X}, d)$ είναι πλήρης και από το Λήμμα 2 γνωρίζουμε ότι η ακολουθία $x_n = \mathbf{T}(x_{n-1})$ είναι Cauchy συμπεραίνουμε ότι η $\{x_n\}$ θα είναι και συγκλίνουσα στον $\mathbf{X}$. Έστω $x^* \in \mathbf{X}$ το όριό της. Θα δείξουμε ότι αυτό αποτελεί σταθερό σημείο της συστολής $\mathbf{T}$. Πράγματι

$$x^* = \lim_{n\to\infty} x_n = \lim_{n\to\infty} \mathbf{T}(x_{n-1}) = \mathbf{T}(\lim_{n\to\infty} x_{n-1}) = \mathbf{T}(x^*)$$

όπου το όριο επιτρέπεται να μπει μέσα στην συνάρτηση $\mathbf{T}$ εφόσον αυτή είναι συστολή και άρα ομοιόμορφα συνεχής και άρα συνεχής. Το μόνο που μένει τώρα είναι να δείξουμε την μοναδικότητα. Έστω ότι η συστολή $\mathbf{T}$ έχει δύο σταθερά σημεία, τα x_1^* και x_2^*. Τότε

$$d(\mathbf{T}(x_1^*), \mathbf{T}(x_2^*)) = d((x_1^*), (x_2^*))$$

όμως γνωρίζουμε ότι η $\mathbf{T}$ είναι συστολή και άρα εφόσον η σταθερά που πολλαπλασιάζει το $d((x_1^*), (x_2^*))$ είναι η μονάδα $1 \notin [0, 1)$ καταλήγουμε σε άτοπο. Άρα το σταθερό σημείο της $\mathbf{T}$ είναι μοναδικό και η απόδειξη έχει ολοκληρωθεί.

10.2 Απόδειξη των αναδρομικών εξισώσεων βελτιστοποίησης Bellman

Η βάση της απόδειξης των εξισώσεων του Bellman είναι η σχέση:

$$V^*(s) = \max_{\alpha \in \mathcal{A}(s)} Q^{\pi^*}(s, \alpha)$$

Γνωρίζουμε όμως ότι, εξ'ορισμού της συνάρτησης Q^π, ισχύει $Q^{\pi^*}(s, \alpha) = \mathbb{E}_{\pi^*}[G_t | s_t = s, \alpha_t = \alpha]$. Συνεπώς

$$V^*(s) = \max_{\alpha \in \mathcal{A}(s)} \mathbb{E}_{\pi^*}[G_t | s_t = s, \alpha_t = \alpha] = \max_{\alpha \in \mathcal{A}(s)} \mathbb{E}_{\pi^*}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | s_t = s, \alpha_t = \alpha\right] =$$

$$\max_{\alpha \in \mathcal{A}(s)} \mathbb{E}_{\pi^*}\left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} | s_t = s, \alpha_t = \alpha\right] = \max_{\alpha \in \mathcal{A}(s)} \mathbb{E}_{\pi^*}[R_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, \alpha_t = \alpha] =$$

$$\max_{\alpha \in \mathcal{A}(s)} \sum_{s'} \mathcal{P}(s, \alpha, s') \left(\mathcal{R}(s, \alpha, s') + \gamma V^*(s')\right)$$

Με παρόμοιους συλλογισμούς και με την βοήθεια της σχέσης (5.1.1) έχουμε για την $Q^*(s, \alpha)$:

$$Q^*(s, \alpha) = \mathbb{E}[R_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, \alpha_t = \alpha] = \mathbb{E}\left[R_{t+1} + \gamma \max_{\alpha' \in \mathcal{A}(s)} Q^*(s_{t+1}, \alpha') | s_t = s, \alpha_t = \alpha\right]$$

$$= \sum_{s'} \mathcal{P}(s, \alpha, s') \left(\mathcal{R}(s, \alpha, s') + \gamma \max_{\alpha' \in \mathcal{A}(s)} Q^*(s', \alpha')\right)$$

και η απόδειξη έχει ολοκληρωθεί.

10.3 Απόδειξη του Θεωρήματος Συμβατότητας Συναρτησιακής Προσέγγισης

Εφόσον το διάνυσμα w ελαχιστοποιεί το μέσο τετραγωνικό σφάλμα, MSE, θα ισχύει ότι $\nabla_w MSE(\theta, w) = 0$. Από τον ορισμό του μέσου τετραγωνικού σφάλματος, ισχύει ότι $\nabla_w \epsilon^2 = 0$. Επίσης από την πρώτη συνθήκη του θεωρήματος προκύπτει ότι $\nabla_w \epsilon(s; \theta, w) = \nabla_\theta \mu_\theta(s)$. Συνεπώς έχουμε

$$\nabla_w MSE(\theta, w) = 0 \Rightarrow$$

$$\mathbb{E}[\nabla_\theta \mu_\theta(s) \epsilon(s; \theta, w)] = 0 \Rightarrow$$

$$\mathbb{E}\left[\nabla_\theta \mu_\theta(s) \nabla_\alpha Q^w(s, \alpha)|_{\alpha = \mu_\theta(s)}\right] = \mathbb{E}\left[\nabla_\theta \mu_\theta(s) \nabla_\alpha Q^\mu(s, \alpha)|_{\alpha = \mu_\theta(s)}\right] =$$

$$\nabla_\theta \mathcal{J}_\beta(\mu_\theta)$$

ή

$$\mathbb{E}\left[\nabla_\theta \mu_\theta(s) \nabla_\alpha Q^\mu(s, \alpha)|_{\alpha = \mu_\theta(s)}\right] = \nabla_\theta \mathcal{J}(\mu_\theta)$$

και η απόδειξη έχει ολοκληρωθεί.

10.4 Απόδειξη των εξισώσεων του αλγορίθμου Back-Propagation

- Απόδειξη της σχέσης $\delta^L = \nabla_\alpha C \odot \sigma'(z^L)$

Η εξίσωση αυτή στην ουσία έχει αποδειχθεί με βάση την συζήτηση που έγινε στο αντίστοιχο κεφάλαιο. Υπενθυμίζουμε λοιπόν ότι

$$\delta_j^L = \frac{\partial C}{\partial z_j^L} = \frac{\partial C}{\partial \alpha_j^L}\frac{\partial \alpha_j^L}{\partial z_j^L} = \frac{\partial C}{\partial \alpha_j^L}\sigma'(z_j^L)$$

εφόσον $\alpha_j^L = \sigma(z_j^L)$

- Απόδειξη της σχέσης $\delta^l = \left[(w^{l+1})^\intercal \delta^{l+1}\right] \odot \sigma'(z^l)$

Αρχίζουμε με τον γνωστό ορισμό $\delta_j^l = \frac{\partial C}{\partial z_j^l}$

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_k^{l+1}}\frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_k \delta_k^{l+1}\frac{\partial z_k^{l+1}}{\partial z_j^l}$$

Γνωρίζουμε όμως ότι

$$z_k^{l+1} = \sum_j w_{kj}^{l+1}\alpha_j^l + b_k^{l+1} = \sum_j w_{kj}^{l+1}\sigma(z_j^l) + b_k^{l+1}$$

Παραγωγίζοντας την παραπάνω σχέση ως προς z_j^l παίρνουμε

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1}\sigma'(z_j^l)$$

Ο όρος που περιλαμβάνει το bias b_k^{l+1} εξαφανίζεται γιατί δεν εξαρτάται από την μεταβλητή z_j^l. Αντικαθιστούμε την έκφραση της μερικής παραγώγου $\frac{\partial z_k^{l+1}}{\partial z_j^l}$ στην σχέση που δίνει το σφάλμα δ_j^l και λαμβάνουμε

$$\delta_j^l = \sum_k w_{kj}^{l+1}\delta_k^{l+1}\sigma'(z_j^l)$$

η οποία είναι ακριβώς η εξίσωση $\delta^l = \left[(w^{l+1})^\intercal \delta^{l+1}\right] \odot \sigma'(z^l)$ γραμμένη σε μορφή συνιστωσών.

- Απόδειξη της σχέσης $\frac{\partial C}{\partial b_j^l} = \delta_j^l$

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} = \delta_j^l \frac{\partial z_j^l}{\partial b_j^l}$$

όμως

$$\frac{\partial}{\partial b_j^l} z_j^l = \frac{\partial}{\partial b_j^l}\left(\sum_k w_{jk}^l \alpha_k^{l-1} + b_j^l\right) = \sum_k \frac{\partial w_{jk}^l \alpha_k^{l-1}}{\partial b_j^l} + \frac{\partial b_j^l}{\partial b_j^l} = 1$$

εφόσον η ποσότητα $w_{jk}^l \alpha_k^{l-1}$ δεν εξαρτάται από το bias b_j^l. Συνεπώς παίρνουμε την επιθυμητή εξίσωση.

- Απόδειξη της σχέσης $\frac{\partial C}{\partial w_{jk}^l} = \alpha_k^{l-1}\delta_j^l$

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{jk}^l} = \delta_j^l \frac{\partial z_j^l}{\partial w_{jk}^l}$$

όμως

$$\frac{\partial z_j^l}{\partial w_{jk}^l} = \frac{\partial}{\partial w_{jk}^l}\left(\sum_k w_{jk}^l \alpha_k^{l-1} + b_j^l\right) = \sum_k \frac{\partial w_{jk}^l \alpha_k^{l-1}}{\partial w_{jk}^l} + \frac{\partial b_j^l}{\partial w_{jk}^l} = \alpha_k^{l-1}$$

εφόσον το bias b_j^l δεν εξαρτάται από την μεταβλητή w_{jk}^l. Συνεπώς παίρνουμε την επιθυμητή εξίσωση και πλέον έχουμε αποδείξει όλες τις εξισώσεις που περιλαμβάνονται στον αλγόριθμο του back-propagation.

Βιβλιογραφία

[1] Timothy P. Lillicrap, Jonathan H. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver and Daan Wierstra. *Continuous Control with Deep Reinforcement Learning*, 2016.

[2] Nils J. Nilsson. *Introduction to Machine Learning*, Robotics Laboratory, Department of Computer Science, Stanford University available at https://book.huihoo.com/pdf/introduction-to-machine-learning/, pp. 1, 1998.

[3] *Introduction to Machine Learning: The Wikipedia Guide*, available at http://www.datascienceassn.org/sites/default/files/Introduction%20to%20Machine%20Learning.pdf, pp. 1-4.

[4] Vincent Francois-Lavet, Riashat Islam, Joelle Pineau, Peter Henderson and Marc G. Bellemare. *An Introduction to Deep Reinforcement learning*, Foundations and Trends in Machine Learning Vol. 11, No 3-4, pp. 15-19, 24-25 2018.

[5] Νταουτίδης Πρόδρομος, Μαστρογεωργόπουλος Σπύρος and Παπαδοπούλου Σημίρα. *Έλεγχος Διεργασιών*, Εκδόσεις ΤΖΙΟΛΑ, pp.1-6, 97-98, 2013.

[6] Ν. Κουμούτσος, Β. Λυγερού and Γ. Μπάφας. *Δυναμική Συστημάτων και Ρύθμιση Διεργασιών*, Εκδόσεις ΕΜΠ, pp. 57-60, 1990.

[7] Kevin Gurney. *An introduction to neural networks*, UCL Press, pp.12-16, 1997.

[8] David Kriesel. *A brief introduction to neural networks*, available at http://www.dkriesel.com, 2007.

[9] Michael A. Nielsen. *Neural Networks and Deep Learning*, available at http://neuralnetworksanddeeplearning.com/, Determination Press, pp. 42-49 2015.

[10] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*, MIT Press, available at http://www.deeplearningbook.org, pp. 226-231, 2016.

[11] Steven Spielberg Pon Kumar.*Deep Reinforcement Learning Approaches for Process Control*, Master Thesis, pp.14-15, 77, 2017.

[12] Yaron Singer. *AM 221: Advanced Optimization, Lecture 9*, pp. 3-5, 2016.

[13] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*, Cambridge University Press, pp. 459-461, 2004.

[14] Balázs Csanád Csáji. *Approximation with Artificial Neural Networks*. MSc Thesis, pp. 11, 2001.

[15] Csaba Szepesvári. *Algorithms for Reinforcement Learning*, Morgan and Claypool, pp. 12-15, 2009.

[16] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction, second edition*, MIT Press, pp. 50-51, 2017.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan, Wierstra and Martin Riedmiller. *Playing Atari with Deep Reinforcement Learning*, Deep Mind Technologies, 2013.

[18] Max Tegmark. *Τι σημαίνει να είσαι άνθρωπος στην εποχή της τεχνητής νοημοσύνης;*, Εκδοτικός Οίκος ΤΡΑΥΛΟΣ, 2018.

[19] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra and Martin Riedmiller. *Deterministic Policy Gradient Algorithms*, 2014.

[20] Xiaoqin Zhang and Huimin Ma. *Pretraining Deep Actor-Critic Reinforcement Learning Algorithms With Expert Demonstrations*, 2018.

[21] Vijay R. Konda and John N. Tsitsiklis. *Actor-Critic Algorithms*, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 2000.

[22] S.P.K. Spielberg, R.B. Gopaluni and P.D. Loewen. *Deep Reinforcement Learning Approaches for Process Control*, 2016.

[23] Steven Spielberg Pon Kumar, Bhushan Gopaluni and Philip Loewen. *Process Control using Deep Reinforcement Learning*, 2017

[24] Tom Schaul, John Quan, Ioannis Antonoglou and David Silver. *Prioritized Experience Replay*, 2016.

[25] Simon Ramstedt. *Deep Reinforcement Learning for Continuous Control*, Bachelor-Thesis, 2016.

[26] Uhlenbeck, George E and Ornstein, Leonard S. *On the theory of the brownian motion.* Physical review, 36(5):823, 1930.

[27] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariance Shift*, 2015.

[28] Matthew Hausknecht and Peter Stone. *Deep Reinforcement Learning in Parametrized Action Space*, 2016.

[29] Haralambos Sarimveis and George Bafas. *Fuzzy model predictive control of non-linear processes using genetic algorithms*, 2002.

[30] Diederik P. Kingma and Jimmy Lei Ba. *Adam: A Method for Stochastic Optimization*, 2015.

Ηλεκτρονικοί Σύνδεσμοι

https://github.com/stevenpjg/ddpg-aigym

http://mcneela.github.io/machine_learning/2017/03/21/Universal-Approximation-Theorem.html

https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#kullback-leibler

https://en.wikipedia.org/wiki/Loss_function

https://en.wikipedia.org/wiki/Hadamard_product_(matrices)

https://xingyuzhou.org/blog/notes/strong-convexity

https://en.wikipedia.org/wiki/Convex_function

https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0

https://deeplizard.com/learn/video/Bcuj2fTH4_4

https://pemami4911.github.io/blog/2016/08/21/ddpg-rl.html

https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html

https://towardsdatascience.com/batch-normalization-theory-and-how-to-use-it-with-tensorflow-1892ca0173ad

https://en.wikipedia.org/wiki/L%C3%A9vy%27s_modulus_of_continuity_theorem

http://web.math.ku.dk/~susanne/StatDiff/Overheads1b

https://galton.uchicago.edu/~lalley/Courses/313/BrownianMotionCurrent.pdf

https://en.wikipedia.org/wiki/Ornstein\T1\textendashUhlenbeck_process

https://www.math.utah.edu/~davar/ps-pdf-files/donsker.pdf

https://en.wikipedia.org/wiki/Stochastic_gradient_descent

https://ai-mrkogao.github.io/reinforcement%20learning/ActorCriticTensorflow/

https://spinningup.openai.com/en/latest/algorithms/ddpg.html

https://www.youtube.com/watch?v=n9K3zJNf75Q