ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ»


# Ανίχνευση Επιθέσεων Άρνησης Εξυπηρέτησης με χρήση Συστημάτων Hierarchical Temporal Memory

Μεταπτυχιακή Διπλωματική Εργασία

του

**Στούμπου Εμμανουήλ**

**Επιβλέπων:** Ομότιμος Καθηγητής Βασίλειος Μάγκλαρης

**Συνεπιβλέπων:** Υποψήφιος Διδάκτορας Μαρίνος Δημολιάνης

Αθήνα, Απρίλιος 2021

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ»


# Ανίχνευση Επιθέσεων Άρνησης Εξυπηρέτησης με χρήση Συστημάτων Hierarchical Temporal Memory

Μεταπτυχιακή Διπλωματική Εργασία

του

**Στούμπου Εμμανουήλ**

**Επιβλέπων:** Ομότιμος Καθηγητής Βασίλειος Μάγκλαρης

**Συνεπιβλέπων:** Υποψήφιος Διδάκτορας Μαρίνος Δημολιάνης

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 27η Απριλίου 2021.

| (Υπογραφή) | (Υπογραφή) | (Υπογραφή) |
|---|---|---|
| ..... | ..... | ..... |
| Βασίλειος Μάγκλαρης | Γεώργιος Στάμου | Ευστάθιος Συκάς |
| Ομότιμος Καθηγητής | Αναπληρωτής Καθηγητής | Καθηγητής |

Αθήνα, Απρίλιος 2021

(Υπογραφή)

.....
Στούμπος Εμμανουήλ

## Περίληψη

Η ασφάλεια δικτύων παραμένει έως και σήμερα μία από τις σημαντικότερες περιοχές έρευνας στον κλάδο της Πληροφορικής. Από απλοί χρήστες ως και παγκόσμιοι οργανισμοί, όλοι τους μπορούν να αποτελέσουν θύματα διαδικτυακών επιθέσεων, υποκλέπτοντας προσωπικές τους πληροφορίες και βλάπτοντας το όνομά τους στον χώρο. Μολονότι υπάρχει ήδη ένα ευρύ φάσμα εργαλείων τα οποία βρίσκουν εφαρμογή στην αντιμετώπιση αυτού του είδους επιθέσεων, νέες τεχνικές συνεχίζουν να αξιοποιούνται καθημερινά ως προς την αποτελεσματική λύση του προβλήματος. Την τελευταία δεκαετία έχει παρατηρηθεί η άνθιση της περιοχής της Μηχανικής Μάθησης, ένα σύνολο αλγορίθμων οι οποίοι μέσω της εκπαίδευσής τους είναι ικανοί να ανακαλύπτουν υποβόσκοντα μοτίβα μεταξύ τεράστιων όγκων δεδομένων, τα οποία μπορούμε μετέπειτα να εκμεταλλευτούμε προς όφελός μας. Η εργασία αυτή ερευνά κατά πόσο είναι εφικτή η χρήση αυτών των αλγορίθμων με σκοπό την ανίχνευση ενός συγκεκριμένου τύπου διαδικτυακών επιθέσεων, γνωστές και ως Κατανεμημένες Επιθέσεις Άρνησης Εξυπηρέτησης. Πιο συγκεκριμένα, ως προς την επίτευξη αυτού του σκοπού αξιοποιούμε τα Τεχνητά Νευρωνικά Δίκτυα, ένα αρκετά διαδεδομένο σύνολο αλγορίθμων Μηχανικής Μάθησης, καθώς και τα συστήματα Hierarchical Temporal Memory, ένα σχετικά νέο αλλά πολλά υποσχόμενο μοντέλο εμπνευσμένο από τον νεοφλοιό, ένα σημαντικό τμήμα του ανθρώπινου εγκεφάλου, υπεύθυνο για ένα πλήθος βασικών λειτουργιών. Μέσω διάφορων μεθόδων εφαρμογής των παραπάνω μοντέλων επιχειρούμε συνεπώς να αντιμετωπίσουμε το πρόβλημα της ανίχνευσης της κακόβουλης δικτυακής κυκλοφορίας, ενώ παράλληλα εστιάζουμε την προσοχή μας στη μελέτη και στην κατανόηση της λειτουργίας των συστημάτων Hierarchical Temporal Memory.



**Λέξεις-κλειδιά:** Μηχανική Μάθηση, Ασφάλεια Δικτύων, Τεχνητά Νευρωνικά Δίκτυα, Αναδρομικά Νευρωνικά Δίκτυα, Αυτοκωδικοποιητές, Συστήματα Hierarchical Temporal Memory, Επιθέσεις DDoS.

**Abstract**

Network security remains to this day an extremely important field of study in Informatics. From everyday users to global organizations, each and every one of them can fall victim to cyber attacks, stealing their personal information and damaging their reputation in the process. While there already exists a wide range of tools used to combat this type of attacks, brand new techniques are being applied every day in order to effectively solve the problem. The last decade has seen the flourishing of Machine Learning, a set of algorithms capable of being trained so that they are able to successfully discover underlying patterns in enormous chunks of data, which can then be used to our advantage. Throughout this work we invistigate the extend to which these algorithms are capable of being used in order to identify a certain type of cyber attacks, namely Distributed Denial-of-Service Attacks. More specifically, we deploy Artificial Neural Networks, a well known set of various algorithms in Machine Learning, as well as Hierarchical Temporal Memory systems, a relatively new but promising model inspired by the neocortex, an important part of the human brain, responsible for a multitude of basic functions. Through implementing said models in a number of ways we therefore attempt to successfully detect malicious network traffic, while also focusing our attention on the study and understanding of Hierarchical Temporal Memory systems.

**Keywords:** Machine Learning, Network Security, Artificial Neural Networks, Recurrent Neural Networks, Autoencoders, Hierarchical Temporal Memory Systems, DDoS Attacks.

# Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον κ. Βασίλειο Μάγκλαρη για την εμπιστοσύνη που μου έδειξε αναθέτοντάς μου αυτήν την εργασία, καθώς επίσης και τον υποψήφιο διδάκτορα Μαρίνο Δημολιάνη ο οποίος ήταν πάντοτε εκεί για να με καθοδηγεί, λύνοντάς μου κάθε απορία με ευχαρίστηση.

# Περιεχόμενα

# Κατάλογος Σχημάτων

# Κατάλογος Πινάκων

# Κεφάλαιο 1

# Εισαγωγή

Η εργασία αυτή αποβλέπει στην επίλυση του προβλήματος της αναγνώρισης του είδους της διαδικτυακής κυκλοφορίας, συγκεκριμένα του εντοπισμού κακόβουλης κίνησης στα πλαίσια της ανίχνευσης διαφορετικών ειδών διαδικτυακών επιθέσεων άρνησης εξυπηρέτησης, χρησιμοποιώντας αλγορίθμους άμεσα προερχόμενους από την περιοχή της Μηχανικής Μάθησης (Machine Learning), η οποία τα τελευταία χρόνια έχει βρει εφαρμογή σε ένα τεράστιο εύρος προβλημάτων, από την αναγνώριση φωνής έως και την κατασκευή αυτόνομων οχημάτων. Πιο συγκεκριμένα, στο εισαγωγικό κεφάλαιο της εργασίας αναφερόμαστε συνοπτικά τόσο στον κλάδο της Μηχανικής Μάθησης ως ένα ενιαίο σύνολο προβλημάτων, όσο και στην φύση και τους κινδύνους των διαδικτυακών επιθέσεων άρνησης εξυπηρέτησης, ενώ στο δεύτερο κεφάλαιο εμβαθύνουμε στην λειτουργία των ευρέως διαδεδομένων μοντέλων των Τεχνητών Νευρωνικών Δικτύων (Artificial Neural Networks), καθώς και των σχετικά νέων αλλά πολλά υποσχόμενων συστημάτων Hierarchical Temporal Memory. Το τρίτο κεφάλαιο αφορά την περιγραφή των μεθόδων βάσει των οποίων επιχειρούμε να εφαρμόσουμε τα δύο παραπάνω είδη μοντέλων με σκοπό την αντιμετώπιση του προβλήματος προς επίλυση, εξετάζοντας παράλληλα παρόμοιες σύγχρονες μεθόδους. Τέλος, στα δύο τελευταία κεφάλαια της εργασίας παρουσιάζονται τα αποτελέσματα των πειραμάτων που εκτελέστηκαν στα πλαίσια της επίλυσης του προβλήματος, καθώς και τα όσα συμπεράσματα εξήχθησαν κατά την εκπόνηση της εργασίας.

## 1.1 Ο Κλάδος της Μηχανικής Μάθησης

Η βασική αρχή η οποία αποτελεί τον πυρήνα κάθε μοντέλου Μηχανικής Μάθησης αφορά την ανίχνευση καθώς και την κατανόηση των γενικότερων μοτίβων, τα οποία είναι ικανά να περιγράψουν την στατιστική φύση των δεδομένων μέσα από τα οποία αναδύονται. Είναι προφανές πως μία τέτοιου είδους προσέγγιση του προβλήματος τις περισσότερες φορές απαιτεί ένα πολυπληθές σύνολο δεδομένων, καθώς αποδεικνύεται πως ο εντοπισμός αυτών των μοτίβων σε μικρές κλίμακες

είναι μία αρκετά δύσκολη έως και αδύνατη διαδικασία.

Οι δύο[1] κυριότερες κατηγορίες στις οποίες διαχωρίζουμε τα είδη προβλημάτων Μηχανικής Μάθησης, είναι τα προβλήματα «Επιβλεπόμενης Μάθησης» (Supervised Learning) και τα προβλήματα «Μη-Επιβλεπόμενης Μάθησης» (Unsupervised Learning). Στην πρώτη κατηγορία ανήκει και το είδος του προβλήματος το οποίο επιχειρούμε να αντιμετωπίσουμε, δηλαδή το πρόβλημα της Ταξινόμησης (Classification), το οποίο αφορά την ορθή κατηγοριοποίηση των δεδομένων στην κλάση που ανήκουν. Για παράδειγμα στα πλαίσια της αναγνώρισης του είδους της διαδικτυακής κίνησης μπορούμε να ορίσουμε δύο διαφορετικές κλάσεις, την «καλόβουλη» και την «κακόβουλη» κίνηση. Αποσπώντας ένα σύνολο δεδομένων ικανό να περιγράψει και τα δύο είδη κίνησης, για παράδειγμα συλλέγοντας πακέτα μεταφοράς δεδομένων τα οποία αφορούν τόσο την καλόβουλη όσο και την κακόβουλη κίνηση, θα θέλαμε να εκπαιδεύσουμε ένα μοντέλο έτσο ώστε μετέπειτα να είναι ικανό να αναγνωρίσει με ποιά από τις δύο αυτές κατηγορίες σχετίζεται ένα οποιοδήποτε νέο άγνωστο πακέτο.

Αυτό το οποίο διαχωρίζει τα προβλήματα Επιβλεπόμενης Μάθησης από τα προβλήματα Μη-Επιβλεπόμενης Μάθησης είναι η ύπαρξη της πληροφορίας η οποία υποδεικνύει σε ποιά από τις επιμέρους κλάσεις ανήκει κάθε στοιχείο του «συνόλου εκπαίδευσης», δηλαδή του συνόλου δεδομένων βασει του οποίου το μοντέλο μας εκπαιδεύεται. Η πληροφορία αυτή μας παρέχεται μέσω ενός πλήθους «ετικετών» (labels), οι οποίες αντιστοιχούν σε κάθε ένα από τα στοιχεία του συνόλου και εκφράζουν την κλάση στην οποία αυτά ανήκουν. Συνεπώς στα πλαίσια της Επιβλεπόμενης Μάθησης ένα σύνολο εκπαίδευσης $S$ αποτελεί μία συλλογή αντικειμένων $(\boldsymbol{x}, y)$, όπου $\boldsymbol{x}$ το διάνυσμα το οποίο περιγράφει τα χαρακτηριστικά του εκάστοτε στοιχείου, και $y$ η αντίστοιχη ετικέτα του. Να τονίσουμε επίσης ότι θα πρέπει πάντοτε να έχουμε στην κατοχή μας ένα ξεχωριστό, συνήθως μικρότερο, σύνολο δεδομένων $S'$, το οποίο ονομάζουμε «σύνολο αξιολόγησης», με σκοπό την μετέπειτα αξιολόγηση της επίδοσης του πλέον εκπαιδευμένου μοντέλου προτού αυτό βρει εφαρμογή σε πραγματικά προβλήματα.

Αν και η εκπαίδευση των μοντέλων είναι σαφώς ευκολότερη όταν οι αντίστοιχες ετικέτες των δεδομένων βρίσκονται στην κατοχή μας, τις περισσότερες φορές η κατασκευή τέτοιων συνόλων αποδεικνύεται μία σχετικά δύσκολη και χρονοβόρα διαδικασία, καθώς ενώ η συλλογή των ίδιων των δεδομένων αποτελεί πλέον μία αρκετά εύκολη υπόθεση, ταυτόχρονα δεν υπάρχει κάποια αυτοματοποιημένη μέθοδος για την εκ των προτέρων αντιστοίχισή τους στην πραγματική τους κλάση. Όταν λοιπόν το σύνολο δεδομένων δεν περιέχει τις ανάλογες ετικέτες των στοιχείων του, τότε κάνουμε λόγο για προβλήματα Μη-Επιβλεπόμενης Μάθησης, τα οποία με την σειρά τους μπορούν να διαχωριστούν σε επιμέρους κατηγορίες προβλημάτων. Παρόλο που η εργασία αυτή αποσκοπά κυρίως στην επίλυση του προβλήματος της Ταξινόμησης, όπως θα δούμε παρακάτω θα εξετάσουμε παράλληλα και την περίπτωση στην οποία δεν έχουμε πρόσβαση στις αντίστοιχες ετικέτες των δεδομένων.

[1] Στην πραγματικότητα υπάρχει και μία τρίτη βασική κατηγορία μάθησης, η «Ενισχυτική Μάθηση» (Reinforcement Learning), η οποία όμως διαφέρει σε αρκετά μεγάλο βαθμό από τις άλλες δύο και δεν μας απασχολεί στα πλαίσια αυτής της εργασίας.

## 1.2 Οι Κατανεμημένες Επιθέσεις Άρνησης Εξυπηρέτησης

Είναι δεδομένο πως με την πρόοδο της τεχνολογίας η διαδικτυακή επικοινωνία μεταξύ συσκευών γίνεται ολοένα και πιο ασφαλής. Παρόλα αυτά δεν παύουν συνεχώς να ανακαλύπτονται νέες μέθοδοι με σκοπό την εκμετάλλευση τυχόν αδύναμων σημείων που ενδέχεται να παρουσιάζουν ορισμένες εφαρμογές του παγκόσμιου ιστού, καθιστώντας κατά αυτόν τον τρόπο την διαδικτυακή πλοήγηση για τον μέσο χρήστη επικίνδυνη, ενώ ταυτόχρονα απειλούν την ορθή λειτουργία των διακομιστών μεγάλων οργανισμών και εταιριών, κάτι το οποίο ενδέχεται να βλάψει το όνομά τους στον χώρο, ή ακόμη και να οδηγήσει σε σημαντική μείωση των εσόδων τους. Για αυτόν τον λόγο, η έρευνα που πραγματοποιείται γύρω από την περιοχή της ασφάλειας δικτύων παραμένει έως και σήμερα μείζονος σημασίας.

Ένα μεγάλο μέρος αυτής της έρευνας σχετίζεται με την ανίχνευση και την επιτυχή αντιμετώπιση διαδικτυακών επιθέσεων. Οι περισσότερες επιθέσεις τέτοιου τύπου αποσκοπούν είτε στην υποκλοπή των προσωπικών πληροφοριών των απλών χρηστών, είτε στην διατάραξη της ομαλής λειτουργίας διακομιστών, συνήθως αυτών οι οποίοι βρίσκονται υπό την ιδιοκτησία μεγάλων εταιριών και οργανισμών κάθε είδους. Στην δεύτερη αυτή κατηγορία ανήκουν και οι «Επιθέσεις Άρνησης Εξυπηρέτησης» (Denial-of-Service Attacks), μέσω των οποίων ο επιτιθέμενος επιχειρεί να κατακλύσει τον διακομιστή-θύμα με υπερβολικά μεγάλη κίνηση, επιβάλλοντάς του να διαχειριστεί ένα τεράστιο πλήθος αιτημάτων, καθιστώντας τον έτσι ανίκανο να ανταπεξέλθει στα αιτήματα των πραγματικών πελατών, ή ακόμη και οδηγώντας στην πλήρη κατάρρευσή του. Όταν ο επιτιθέμενος αξιοποιεί πολλές διαφορετικές συσκευές, με διαφορετικές συνήθως IP διευθύνσεις, για να επιτύχει τον σκοπό του, τότε κάνουμε λόγο για «Κατανεμημένες Επιθέσεις Άρνησης Εξυπηρέτησης» (Distributed Denial-of-Service Attacks), γνωστές και ως επιθέσεις DDoS.

Ένα ευρέως γνωστό είδος DDoS επιθέσεων το οποίο επιχειρούμε να αντιμετωπίσουμε σε αυτή την εργασία αποτελούν οι λεγόμενες DNS Amplification επιθέσεις, μέσω των οποίων οι επιτιθέμενοι εκμεταλλεύονται τις υπηρεσίες που παρέχουν οι διακομιστές DNS με σκοπό την κατανάλωση του εύρους ζώνης των διακομιστών του θύματος. Οι διακομιστές αυτοί αποτελούν ένα αναπόσπαστο κομμάτι του διαδικτύου καθώς είναι υπεύθυνοι για την μετατροπή του ονόματος τομέα κάθε ιστοσελίδας στην αντίστοιχη διεύθυνση IP. Κάθε φορά που ο μέσος χρήστης πληκτρολογεί στον περιηγητή το όνομα τομέα της ιστοσελίδας που επιθυμεί να επισκεπτεί, στην ουσία υποβάλλει ένα ερώτημα σε έναν ή περισσότερους διακομιστές DNS έτσι ώστε να του παρέχουν την πραγματική διεύθυνση της ιστοσελίδας. Οι επιτιθέμενοι εκμεταλλεύονται το σύστημα αυτό, εκτελώντας μέσω μίας η περισσότερων συσκευών ένα μεγάλο πλήθος ερτημάτων αυτού του είδους, θέτοντας παράλληλα την πραγματική διεύθυνση IP του θύματος, την οποία γνωρίζουν εκ των προτέρων, ως την διεύθυνση παραλήπτη. Οι διακομιστές του θύματος επομένως δέχονται ξαφνικά από τους διακομιστές DNS ένα μεγάλο πλήθος απαντήσεων σε ερωτήματα τα οποία οι ίδιοι δεν έχουν υποβάλλει, με τελικό αποτέλεσμα την συμφόρηση της κυκλοφορίας τους.

Ούτως ώστε να καταναλώσουν όσο το δυνατόν μεγαλύτερο μέρος του εύρους ζώνης του θύματος, οι επιτιθέμενοι φροντίζουν έτσι ώστε οι απαντήσεις στα ερωτήματα που εκτελούν να είναι όσο πιο «ογκώδεις» γίνεται. Αυτό συνήθως πραγματοποιείται υποβάλλοντας ερωτήματα τα οποία ζητούν επιπλέον δευτερεύουσες πληροφορίες, ή ακόμη και μέσω της εκμετάλλευσης ορισμένων ιστοσελίδων, κακόβουλων και μη, οι οποίες είναι κατασκευασμένες κατά τέτοιο τρόπο έτσι ώστε να παράγουν υπερβολικά ογκώδη πακέτα δεδομένων.

Εκτός από το είδος των επιθέσεων που περιγράφεται παραπάνω, οι οποίες αξιοποιούν κυρίως τα διαδικτυακά πρωτόκολλα DNS και UDP για την εκτέλεσή τους, σε αυτή την εργασία επιχειρούμε να αντιμετωπίσουμε ένα μεγαλύτερο σύνολο επιθέσεων, κάθε μία από τις οποίες εκμεταλλεύεται διαφορετικά πρωτόκολλα και εφαρμογές του διαδικτύου. Παρόλα αυτά οι επιθέσεις με τις οποίες θα ασχοληθούμε σε αυτην την εργασία, όντας όλες τους στην ουσία επιθέσεις τύπου DDoS, μοιράζονται έναν κοινό τελικό σκοπό, δηλαδή την συμφόρηση της διαδικτυακής κυκλοφορίας από και προς τους διακομιστές του θύματος. Συνεπώς μας δίνεται η δυνατότητα να εξετάσουμε όχι μονάχα την ικανότητα των μοντέλων να ανιχνεύσουν μία επίθεση, αλλά παράλληλα και κατά πόσο είναι εφικτό για ένα μοντέλο το οποίο έχει εκπαιδευτεί βάσει δεδομένων που αντιστοιχούν σε μία συγκεκριμένη διαδικτυακή επίθεση A, να είναι ικανό να αναγνωρίσει επιτυχώς μία ελαφρώς διαφορετική επίθεση B.

# Κεφάλαιο 2

# Θεωρητικό Υπόβαθρο

## 2.1 Τα Τεχνητά Νευρωνικά Δίκτυα

Είναι ευρέως γνωστό πως τα τελευταία χρόνια τα μοντέλα των νευρωνικών δικτύων αποτελούν την κύρια επιλογή όσο αφορά την αντιμετώπιση μίας πληθώρας προβλημάτων στην περιοχή της Μηχανικής Μάθησης. Λόγω της περίπλοκης λειτουργίας τους, καθώς και της ύπαρξης ενός μεγάλου πλήθους διαφορετικών αρχιτεκτονικών, δεν είναι δύσκολο να αποδεχτούμε το γεγονός ότι τα μοντέλα αυτά αποτελούν προϊόν μίας συνεχούς και μακρόχρονης έρευνας που εκτείνεται σε διάστημα κάμποσων δεκαετιών, μέσω της οποίας έχουμε οδηγηθεί στις διάφορες κατηγορίες δικτύων που γνωρίζουμε σήμερα. Από τα «Συνελικτικά Νευρωνικά Δίκτυα» (Convolutional Neural Networks) τα οποία είναι κατάλληλα για την επεξεργασία εικόνων έως και τα μοντέλα «Long Short-Term Memory» που χρησιμοποιούνται κυρίως σε συνδυασμό με χρονομετάβλητα δεδομένα, τα νευρωνικά δίκτυα φαίνεται να βρίσκουν εφαρμογή σχεδόν σε κάθε είδος προβλήματος. Παρακάτω θα εξετάσουμε τα θεμέλια των μοντέλων αυτών, επιχειρώντας παράλληλα να ρίξουμε φως στον τρόπο λειτουργίας τους.

### 2.1.1 Ο Αλγόριθμος Perceptron

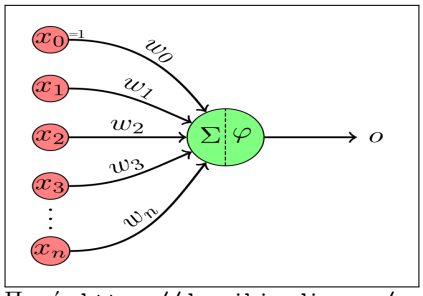
Μπορεί κανείς να δηλώσει χωρίς δεύτερη σκέψη, ότι η δημιουργία του αλγορίθμου «Perceptron» αποτελεί την αφετηρία των τεχνητών νευρωνικών δικτύων, καθώς σηματοδοτεί την έναρξη της ερευνητικής πορείας η οποία μας έχει οδηγήσει στα μοντέλα του σήμερα. Εφευρημένος κατά τα τέλη της δεκαετίας του 50′ στο Αεροναυτικό Εργαστήριο του Κορνέλλ από τον Φράνκ Ρόσενμπλατ [1], ο αλγόριθμος αυτός αφορά την κατασκευή ενός απλού γραμμικού δυαδικού ταξινόμητη. Λέγοντας «δυαδικός» εννοούμε ότι ο αλγόριθμος είναι ικανός να διαχωρίσει στοιχεία δύο και μόνο διαφορετικών κλάσεων, ενώ με τον όρο «γραμμικός» αναφερόμαστε στο γεγονός ότι το μοντέλο αναπαριστάνεται ως ένα υπερεπίπεδο στον διανυσματικό χώρο των διαστάσεων των δεδομένων. Για παράδειγμα, στον χωρο των δύο

διαστάσεων το μοντέλο αυτό αποτελεί μονάχα μία ευθεία. Στον χώρο των τριών διαστάσεων, η αναπαράστασή του γίνεται μέσω ενός επιπέδου, και ούτω καθεξής. Για αυτόν τον λόγο, ο αλγόριθμος Perceptron βρίσκει εφαρμογή μονάχα σε περιπτώσεις όπου τα δεδομένα είναι γραμμικώς διαχωρίσιμα, που σημαίνει ότι ένα απλό υπερεπίπεδο είναι αρκετό ώστε να τα διαχωρίσει πλήρως.

Εξετάζοντας τον αλγόριθμο πλησιέστερα θα παρατηρήσουμε ότι η λειτουργία του είναι αρκετά απλή και εύκολα κατανοητή. Δοθέντος ενός συνόλου εκπαίδευσης αποτελούμενο από ένα πλήθος πλειάδων $(\boldsymbol{x}_i, y_i)$, όπου $\boldsymbol{x}_i \in \mathbb{R}^d$ το διάνυσμα χαρακτηριστικών του στοιχείου, και $y_i = \pm 1$ η τιμή που υποδεικνύει την κλάση στην οποία αυτό ανήκει, ο αλγόριθμος εξάγει ένα διάνυσμα $\boldsymbol{w} \in \mathbb{R}^{d+1}$, μέσω του οποίου στην ουσία ορίζεται ένα υπερεπίπεδο στον διανυσματικό χώρο $\mathbb{R}^d$, ικανό να διαχωρίσει τα δεδομένα στις δύο επιμέρους κλάσεις. Σε αυτό το σημείο θα πρέπει να τονίσουμε ότι το διάνυσμα $\boldsymbol{w}$ δεν περιέχει $d$, αλλά $d+1$ τιμές. Αυτό οφείλεται στο γεγονός ότι στην πρώτη θέση $w_0$ του διανύσματος, αποθηκεύουμε μία επιπλέον τιμή την οποία αποκαλούμε «bias», και η οποία είναι απαραίτητη για την ορθή λειτουργία του μοντέλου. Εάν για παράδειγμα αναλογιστούμε τον χώρο των δύο διαστάσεων, για τον οποίο θα ισχύει $\boldsymbol{w} \in \mathbb{R}^3$, τότε το μοντέλο αναπαριστάνεται ως μία ευθεία, της οποίας η κλίση ορίζεται από τον λόγο των τιμών $w_1$ και $w_2$. Όσο αφορά το σημείο στο οποίο η ευθεία τέμνει τον κατακόρυφο άξονα $y$, η πληροφορία αυτή μας δίνεται από το bias $w_0$. Συνεπώς καταλήγουμε ότι η ύπαρξη αυτού του όρου είναι απαραίτητη προϋπόθεση για την κατασκευή ενός ευέλικτου μοντέλου. Για το υπόλοιπο αυτής της εργασίας, θα θεωρήσουμε ότι ο όρος bias είναι πάντοτε ενσωματωμένος στο αντίστοιχο διάνυσμα βαρών, ενώ για κάθε διάνυσμα εισόδου $\boldsymbol{x}$ θα ορίζεται αντίστοιχα μία επιπλέον τιμή $x_0 = 1$, με σκοπό την ορθή εκτέλεση των πράξεων.

Πως όμως ο αλγόριθμος μαθαίνει από τα δεδομένα έτσι ώστε να εξάγει το τελικό διάνυσμα $\boldsymbol{w}$; Όπως γίνεται και με τα περισσότερα μοντέλα Μηχανικής Μάθησης, έτσι και το μοντέλο Perceptron εκπαιδεύεται σαρώνοντας επαναληπτικά τα στοιχεία του συνόλου εκπαίδευσης. Δεδομένου ενός διανύσματος $\boldsymbol{x} \in \mathbb{R}^d$ στην είσοδο, η τιμή εξόδου $o$ του μοντέλου υπολογίζεται ως εξής: Αρχικά υπολογίζουμε το εσωτερικό γινόμενο των διανυσμάτων $\boldsymbol{x}$ και $\boldsymbol{w}$. Στην συνέχεια, το αποτέλεσμα της πράξης αυτής περνάει μέσα από μία συνάρτηση $\phi$, την οποία αποκαλούμε «συνάρτηση ενεργοποίησης» (activation function), έτσι ώστε να λάβουμε την τελική έξοδο. Στην περίπτωση του αλγορίθμου Perceptron, ως συνάρτηση ενεργοποίησης $\phi$ χρησιμοποιούμε πάντοτε την εξής βηματική συνάρτηση:

Σχήμα 2.1: Το μοντέλο Perceptron.




Πηγή: `https://de.wikipedia.org/wiki/Datei:Perceptron-unit.svg`

$$\phi(x) = \begin{cases} +1\ , & x \geq 0 \\ -1\ , & x < 0 \end{cases}$$

Η παραπάνω διαδικασία υπολογισμού της τιμής εξόδου $o$ του μοντέλου μπορεί να εκφραστεί μαθηματικά μέσω του παρακάτω τύπου:

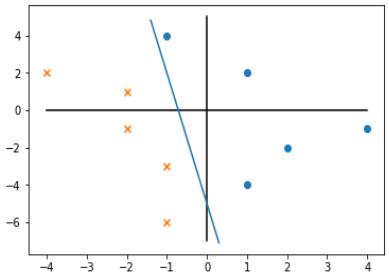
$$o = \phi(\langle \boldsymbol{w}, \boldsymbol{x} \rangle) = \phi\left(w_0 + \sum_{j=1}^{d} w_j \cdot x_j\right)$$

Θα πρέπει έως τώρα να είναι εμφανές, ότι η έξοδος $o$ λαμβάνει πάντοτε μία εκ των δύο τιμών $+1$ ή $-1$. Βάσει αυτού μπορούμε να ισχυριστούμε ότι η έξοδος του μοντέλου αποτελεί ουσιαστικά μία πρόβλεψη της κλάσης στην οποία ανήκει το εκάστοτε στοιχείο εισόδου. Έστω τώρα ότι κατά την επανάληψη $t$, ο αλγόριθμος δέχεται μία πλειάδα $(\boldsymbol{x}^{(t)}, y^{(t)})$ βάσει της οποίας παράγει την έξοδο $o^{(t)}$. Εάν ισχύει ότι $o^{(t)} = y^{(t)}$, τότε αυτό σημαίνει ότι το μοντέλο προέβλεψε σωστά την κλάση του στοιχείου, με αποτέλεσμα το διάνυσμα βαρών να μην χρήζει τροποποίησης. Στην περίπτωση όμως που ισχύει ότι $o^{(t)} \neq y^{(t)}$, τότε το εκάστοτε διάνυσμα βαρών $\boldsymbol{w}^{(t)}$ επαναϋπολογίζεται βάσει του εξής τύπου:

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \beta \cdot y^{(t)} \cdot \boldsymbol{x}'^{(t)}$$

Με τον όρο $\beta$ συμβολίζουμε τον «ρυθμό μάθησης» (learning rate). Η τιμή αυτή, η οποία συνήθως ανήκει στο διάστημα $[0, 1]$, ουσιαστικά καθορίζει το μέγεθος των μεταβολών των βαρών του μοντέλου σε κάθε επανάληψη. Μικρές τιμές του $\beta$ συνεπάγονται αργή αλλά συνάμα σταθερή εκπαίδευση του μοντέλου. Από την άλλη χρησιμοποιώντας έναν υψηλό ρυθμό μάθησης οδηγούμαστε σε μία ναι μεν ταχύτερη αλλά ταυτόχρονα επισφαλής εκπαίδευση, καθώς ένας υψηλός ρυθμός μάθησης ενδέχεται να εισάγει αστάθεια στην διαδικασία εκπαίδευσης.

Μέσω της διαδικασίας που εξηγείται παραπάνω το υπερεπίπεδο το οποίο ορίζεται από το διάνυσμα βαρών $\boldsymbol{w}$ του μοντέλου συνεχώς μετατοπίζεται και περιστρέφεται εντός του διανυσματικού χώρου μέσα στον οποίο βρίσκεται, με τελικό σκοπό να τον διαχωρίσει σε δύο ξεχωριστές περιοχές, στην κάθε μία από τις οποίες θα υπάρχουν τα στοιχεία του συνόλου δεδομένων της αντίστοιχης κλάσης. Θα πρέπει εδώ να τονίσουμε ξανά ότι ούτως ώστε ο αλγόριθμος να καταφέρει εν τέλει να τερματίσει, απαραίτητη προϋπόθεση αποτελεί το γεγονός πως τα στοιχεία του συνόλου εκπαίδευσης θα πρέπει να είναι γραμμικώς διαχωρίσιμα. Τέλος, έχοντας εκπαιδεύσει το μοντέλο ο αλγόριθμος είναι πλέον ικανός να ταξινομήσει ένα νέο άγνωστο στοιχείο εισόδου $\boldsymbol{x}$ μέσω του προσήμου της τιμής που λαμβάνουμε στην έξοδο.

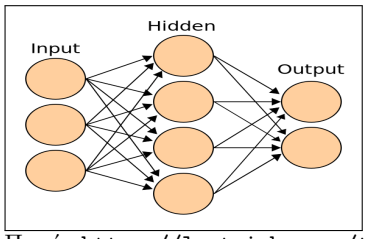



Σχήμα 2.2: Ένα μοντέλο Perceptron στον διανυσματικό χώρο δύο διαστάσεων, το οποίο έχει διαχωρίσει επιτυχώς τις δύο κλάσεις (κουκίδες και σταυρούς).

### 2.1.2 Το Πολυστρωματικό Perceptron

Παρά την επιτυχία του αλγορίθμου Perceptron, το γεγονός ότι το μοντέλο αυτό είναι ανίκανο να διαχειριστεί μη γραμμικώς διαχωρίσιμα δεδομένα, καθώς και ότι περιορίζεται μονάχα σε προβλήματα δυαδικής ταξινόμησης, ώθησε την έρευνα στην αναζήτηση πιο σύνθετων μοντέλων, τα οποία θα μπορούσαν να ξεπεράσουν τέτοιου είδους προβλήματα. Οι τότε ερευνητές σύντομα συνειδητοποίησαν ότι ήταν εφικτό να συνδυάσουν πολλά Perceptron σε ένα ενιαίο ισχυρό μοντέλο, ικανό να αντιμετωπίσει προβλήματα ταξινόμησης πολλών κλάσεων. Ταυτόχρονα διαδόθηκε η χρήση ενός πλήθους διαφορετικών συναρτήσεων ενεργοποίησης, πέρα της βηματικής, μέσω των οποίων θα μπορούσε να αντιμετωπιστεί το πρόβλημα της γραμμικότητας.

Σχήμα 2.3: Ένα Πολυστρωματικό Perceptron ενός κρυφού επιπέδου.




Πηγή: `https://laptrinhx.com/the-deep-history-of-deep-learning-769845960`

Το μοντέλο αυτό σύντομα απέκτησε ένα συγκεκριμένο πρότυπο ως προς την δομή των επιμέρους Perceptron που το αποτελούν, το οποίο συνεχίζει να εφαρμόζεται μέχρι και σήμερα. Λόγω αυτού του προτύπου, το οποίο αντιστοιχεί στην οργάνωση των Perceptron σε ένα πλήθος επιπέδων, ή αλλιώς στρωμάτων, τα οποία συνδέονται μεταξύ τους, το τελικό μοντέλο πήρε το όνομα «Πολυστρωματικό Perceptron»[1] (Multilayer Perceptron). Μπορούμε να πούμε ότι το μοντέλο αυτό αποτέλεσε το πρώτο ολοκληρωμένο νευρωνικό δίκτυο. Αντίστοιχα, τα επιμέρους μοντέλα Perceptron που το αποτελούν, τα αποκαλούμε σήμερα με το όνομα «νευρώνες» του δικτύου. Παρόλα αυτά, η λειτουργία τους παραμένει ακριβώς η ίδια. Παρακάτω παρουσιάζονται τα τρία βασικά επίπεδα οργάνωσης που συνιστούν ένα νευρωνικό δίκτυο:

1. Επίπεδο Εισόδου: Το επίπεδο αυτό είναι μοναδικό και βρίσκεται πάντοτε στην αρχή του δικτύου έτσι ώστε να δεχτεί την είσοδο. Δεν αποτελείται από πραγματικούς νευρώνες, καθώς στο επίπεδο αυτό δεν εκτελείται κάποια πράξη παρά μόνο η προώθηση της εισόδου στο πρώτο κρυφό επίπεδο.

2. Κρυφό Επίπεδο: Η ύπαρξη τουλάχιστον ενός κρυφού επιπέδου είναι αναγκαία προϋπόθεση για την ορθή λειτουργία ενός νευρωνικού δικτύου, καθώς το επίπεδο αυτό είναι υπεύθυνο για το μεγαλύτερο μέρος των υπολογισμών που εκτελούνται πάνω στην είσοδο. Η έξοδος κάθε κρυφού επιπέδου στην ουσία ορίζεται ως ένα διάνυσμα μεγέθους διαστάσεων ίσο με το πλήθος των νευρώνων από τους οποίους το επίπεδο απαρτίζεται, καθώς κάθε ξεχωριστή τιμή του διανύσματος αυτού αντιστοιχεί στην έξοδο ενός και μόνο διαφορετικού νευρώνα του επιπέδου. Στην συνηθέστερη περίπτωση ύπαρξης πολλών

[1] Το όνομα αυτό πλέον χρησιμοποιείται όλο και λιγότερο, καθώς προτιμάται ο όρος «Νευρωνικά Δίκτυα Εμπρόσθιας Διάδοσης».

κρυφών επιπέδων, μπορούμε να τα φανταστούμε τοποθετημένα στην σειρά, το ένα μετά το άλλο, έτσι ώστε το πρώτο κρυφό επίπεδο να προωθεί την έξοδό του στο δεύτερο, αυτό με την σειρά του στο τρίτο και ούτω κάθεξης, μέχρι το τελευταίο επίπεδο να παράγει το διάνυσμα το οποίο αποτελεί την είσοδο στο επίπεδο εξόδου.

3. Επίπεδο Εξόδου: Όπως φανερώνει το όνομά του, το επίπεδο αυτό είναι υπεύθυνο για την παραγωγή της εξόδου ολόκληρου του δικτύου. Το πλήθος των νευρώνων αυτού του επιπέδου ορίζεται βάσει του αριθμού των κλάσεων του προβλήματος προς επίλυση. Έχοντας αντιστοιχήσει κάθε έναν από τους νευρώνες και με μία διαφορετική κλάση, γίνεται πλέον αντιληπτός ο τρόπος ταξινόμησης ενός στοιχείου, καθώς η κλάση που αναπαριστάνεται από τον εκάστοτε νευρώνα του επιπέδου που εξάγει την υψηλότερη τιμή, αποτελεί και την τελική πρόβλεψη του δικτύου για την πραγματική κλάση του εκάστοτε στοιχείου.

Είναι επίσης σημαντικό να τονίσουμε ότι κάθε νευρώνας ενός επιπέδου, εκτός από αυτούς του επιπέδου εισόδου, συνδέεται με κάθε νευρώνα του αμέσως προηγούμενου επιπέδου. Αυτός είναι ο πιο συνηθισμένος τρόπος σύνδεσης των νευρώνων, βάσει του οποίου ορίζεται ένα «Πλήρως Συνδεδεμένο» (Fully Connected) δίκτυο. Οι συνδέσεις αυτές, τις οποίες στα πλαίσια των νευρωνικών δικτύων αποκαλούμε «συνάψεις», κατέχουν ορισμένες τιμές, ή αλλιώς «συναπτικά βάρη». Τα σύνολο όλων των συναπτικών βαρών ενός νευρώνα οργανώνεται σε ένα διάνυσμα το οποίο ονομάζουμε το «διάνυσμα βαρών» του. Όπως συνέβαινε κατά την εκτέλεση του αλγορίθμου Perceptron, έτσι και τώρα κάθε νευρώνας υπολογίζει το εσωτερικό γινόμενο του διανύσματος βαρών με το διάνυσμα εισόδου του, το οποίο στην συνέχεια περνάει μέσα από μία συνάρτηση ενεργοποίησης $\phi$ έτσι ώστε να πάρουμε την τελική έξοδο του νευρώνα.

Μπορούμε λοιπόν να φανταστούμε το νευρωνικό δίκτυο ως έναν σταδιακό μετασχηματισμό του διανύσματος εισόδου. Κάθε κρυφό επίπεδο παράγει μέσω των υπολογισμών που εκτελούν οι νευρώνες του και από ένα νέο διάνυσμα, το οποίο τροφοδοτείται στο αμέσως επόμενο επίπεδο. Η διαδικασία αυτή επαναλαμβάνεται έως ότου να λάβουμε το διάνυσμα εξόδου, βάσει του οποίου πραγματοποιείται η πρόβλεψη του δικτύου σχετικά με την κλάση του εκάστοτε στοιχείου στην είσοδο. Συνεπώς, θα λέγαμε ότι η εκπαίδευση ενός νευρωνικού δικτύου αφορά μονάχα τον υπολογισμό των συναπτικών βαρών του, έτσι ώστε οι προβλέψεις του να αντιστοιχούν όσο το δυνατόν περισσότερο στην πραγματικότητα.

Ωστόσο γίνεται εύκολα κατανοητό ότι η εκπαίδευση ενός ογκώδους δικτύου που αποτελείται από ένα μεγάλο πλήθος κρυφών επιπέδων, με το καθένα από αυτά να περιέχει εκατοντάδες νευρώνες, δεν είναι εύκολη υπόθεση. Αυτό ήταν εξάλλου που εμπόδιζε την πρακτική εφαρμογή των νευρωνικών δικτύων ως προς την επίλυση πραγματικών προβλημάτων. Θα έπρεπε να περάσουν περίπου ακόμη δύο δεκαετίες έως ότου να εφευρεθεί ο αλγόριθμος, ο οποίος θα επέτρεπε στα μοντέλα αυτά να εκπαιδευτούν κάνοντας χρήση ενός συνόλου δεδομένων, μα το σημαντικότερο από όλα, ο νέος αυτός αλγόριθμος θα εισήγαγε μία καινοτόμο διαδικασία εκπαίδευσης, ταχύτερη σε μεγάλο βαθμό από ό,τι ήταν έως τότε εφικτό.

### 2.1.3 Ο Αλγόριθμος Backpropagation

Αν και από πολλούς υποστηρίζεται ότι ο αλγόριθμος «Backpropagation» είχε ήδη επανεφευρεθεί κάμποσες φορές στο παρελθόν, ξεκινώντας από την δεκαετία του 60′, είναι βέβαιο πως η ύπαρξή του γνωστοποιήθηκε στην ευρύτερη επιστημονική κοινότητα κατά τα μέσα της δεκαετίας του 80′ μέσω της δουλειάς των Ρούμελχαρτ, Χίντον, και Γουίλιαμς [2]. Βάσει του εν λόγω αλγορίθμου ορίζονται ουσιαστικά τρία διαφορετικά υπολογιστικά στάδια, τα οποία διέπουν την εκπαίδευση ενός νευρωνικού δικτύου:

1. Στάδιο ανάκλησης: Κατά το πρώτο στάδιο η είσοδος περνάει με στην σειρά μέσα από κάθε επίπεδο του δικτύου έτσι ώστε να λάβουμε το διάνυσμα εξόδου, βάσει του οποίου υπολογίζεται το σφάλμα του δικτύου.

2. Στάδιο οπισθοδιάδοσης του σφάλματος: Βάσει του αλγορίθμου Backpropagation το σφάλμα «μεταφέρεται» πίσω σε όλους τους νευρώνες του δικτύου, ξεκινώντας από το επίπεδο εξόδου φτάνοντας έως και το πρώτο κρυφό επίπεδο, καθορίζοντας έτσι το «μερίδιο ευθύνης» του κάθε νευρώνα για το τελικό σφάλμα.

3. Στάδιο ανανέωσης των συναπτικών βαρών: Βάσει της οπισθοδιάδοσης του σφάλματος ανανεώνουμε τα συναπτικά βάρη κάθε νευρώνα του δικτύου ξεχωριστά.

Το στάδιο της ανάκλησης θα πρέπει έως τώρα να είναι αρκετά κατανοητό καθώς έχουμε ήδη αναφερθεί σε αυτό προηγουμένως. Ουσιαστικά το στάδιο αυτό αφορά μονάχα τον υπολογισμό της εξόδου του δικτύου. Αναφορικά με το δεύτερο στάδιο θα πρέπει αρχικά να αποσαφηνήσουμε τον όρο «σφάλμα» του δικτύου. Για αυτό τον λόγο θα εισάγουμε δύο νέες έννοιες, την «Συνάρτηση Απώλειας» (Loss Function) και την «Συνάρτηση Κόστους» (Cost Function).

Όπως δηλώνει και το όνομά της, η συνάρτηση κόστους δεν είναι τίποτε άλλο παρά μόνο μία συνάρτηση μέσω της οποίας ποσοτικοποιούμε το συνολικό σφάλμα ή αλλιώς «κόστος» του δικτύου, δηλαδή με απλά λόγια την αδυναμία του δικτύου να εκτελεί σωστές προβλέψεις. Ενώ η συνάρτηση κόστους υπολογίζεται βάσει ενός ενιαίου συνόλου δεδομένων, ή πολλές φορές ενός υποσυνόλου αυτού, η συνάρτηση απώλειας από την άλλη υπολογίζεται χρησιμοποιώντας κάθε φορά ένα και μόνο μεμονωμένο στοιχείο του συνόλου. Θα μπορούσαμε λοιπόν να πούμε ότι η συνάρτηση κόστους ορίζεται ως μία πράξη πάνω στις επιμέρους τιμές που υπολογίζει η συνάρτηση απώλειας για κάθε ένα από τα στοιχεία του συνόλου. Για να μην περιπλέξουμε περισσότερο τα πράγματα, θα συνδυάσουμε τις έννοιες των συναρτήσεων απώλειας και κόστους σε μία και μόνο συνάρτηση την οποία θα αποκαλούμε «Συνάρτηση Σφάλματος» (Error Function).

Η συνάρτηση σφάλματος δεν αφορά αυστηρά και μόνο τα νευρωνικά δίκτυα, πράγμα που σημαίνει ότι την συναντάμε σε πάρα πολλά μοντέλα μηχανικής μάθησης. Στο παρακάτω παράδειγμα, θα αναλογιστούμε την συνάρτηση «Μέσου Τετραγωνικού Σφάλματος» (Mean Square Error - MSE), η οποία ορίζεται ως η μέση

τιμή της συνάρτησης τετραγωνικού σφάλματος, μία από τις κύριες επιλογές συνάρτησης ελαχιστοποίησης για την εύρεση των παραμέτρων σε μοντέλα γραμμικής παλινδρόμησης. Δοθέντος λοιπόν ενός συνόλου δεδομένων αποτελούμενο από $n$ πλειάδες $(\boldsymbol{x}_i, y_i)$, η συνάρτηση μέσου τετραγωνικού σφάλματος ορίζεται ως εξής:

$$J(\boldsymbol{e}) = \frac{1}{2n} \sum_{i=1}^{n} e_i^2$$

Το διάνυσμα $\boldsymbol{e}$ περιέχει τα επιμέρους σφάλματα $e_i$, καθένα από τα οποία υπολογίζεται βάσει του αντίστοιχου στοιχείου $(\boldsymbol{x}_i, y_i)$. Πως όμως μπορούμε να υπολογίσουμε τα σφάλματα $e_i$ στα πλαίσια ενός νευρωνικού δικτύου;

Ας αναλογιστούμε αρχικά την έξοδο ενός δικτύου. Έστω ότι καλούμαστε να επιλύσουμε ένα πρόβλημα ταξινόμησης στοιχείων σε $m$ διαφορετικές κλάσεις. Συνεπώς, το διάνυσμα εξόδου θα περιέχει $m$ διαφορετικές τιμές, κάθε μία από τις οποίες θα αντιστοιχεί και σε μία από τις κλάσεις. Έχει καταστεί κοινή πρακτική να μετασχηματίζουμε τις τιμές του διανύσματος εξόδου σε μία κατανομή πιθανοτήτων, τέτοια ώστε το άθροισμά τους να ισούται με την μονάδα. Βάσει αυτού, μπορούμε να ερμηνέυσουμε κάθε τιμή του διανύσματος εξόδου ως την πιθανότητα που το δίκτυο αποδίδει στο εκάστοτε στοιχείο να ανήκει σε κάθε μία από τις κλάσεις. Φαίνεται λογικό άρα να υποθέσουμε ότι για κάποιο στοιχείο $(\boldsymbol{x}_k, y_k)$ το οποίο έστω ότι ανήκει σε μία κλάση $c$, δηλαδή ισχύει πως $y_k = c$, το ιδανικό διάνυσμα εξόδου που θα μπορούσαμε να λάβουμε από το δίκτυο θα ήταν ένα διάνυσμα $\boldsymbol{o}_k$, τέτοιο ώστε η $c$-οστή τιμή του να ισούται με την μονάδα, ενώ όλες οι υπόλοιπες τιμές του να είναι μηδενικές. Με αυτόν τον τρόπο το δίκτυο θα εξέφραζε το γεγονός ότι το στοιχείο $(\boldsymbol{x}_k, y_k)$ ανήκει στην κλάση $c$ με πιθανότητα 1, ενώ η πιθανότητα να ανήκει σε οποιαδήποτε από τις υπόλοιπες κλάσεις είναι μηδενική, κάτι το οποίο ασφαλώς συμπίπτει με την πραγματικότητα.

Επιστρέφοντας πίσω στην συνάρτηση μέσου τετραγωνικού σφάλματος, βάσει της παραπάνω λογικής θα υπολογίσουμε το συνολικό σφάλμα του δικτύου. Για κάθε στοιχείο $(\boldsymbol{x}_i, y_i)$ του συνόλου δεδομένων, αρχικά κατασκευάζουμε τα «ιδανικά» διανύσματα εξόδου $\boldsymbol{d}_i$ ως εξής:

$$d_{ij} = \begin{cases} 1, & j = y_i \\ 0, & j \neq y_i \end{cases} \quad , \quad \forall j \in \{0, 1, ..., m-1\}$$

Στην συνέχεια υπολογίζουμε κάθε σφάλμα $e_i$ βάσει του αντίστοιχου ιδανικού διανυσμάτως $\boldsymbol{d}_i$ μα και του πραγματικού διανύσματος εξόδου του δικτύου $\boldsymbol{o}_i$, ποσοτικοποιώντας έτσι την «απόσταση» που παρεμβάλλεται μεταξύ της πρόβλεψης του δικτύου και της πραγματικότητας. Μέσω της συνάρτησης μέσου τετραγωνικού σφαματος, τα σφάλματα $e_i$ υπολογίζονται ως εξής:

$$e_i = \parallel \boldsymbol{o}_i - \boldsymbol{d}_i \parallel_2 = \sqrt{\sum_{j=0}^{m-1} (o_{ij} - d_{ij})^2}$$

Συνεπώς, το συνολικό σφάλμα του δικτύου υπολογίζεται μέσω του παρακάτω τύπου:

$$J(\boldsymbol{e}) = \frac{1}{2n}\sum_{i=1}^{n} e_i^2 = \frac{1}{2n}\sum_{i=1}^{n} \parallel \boldsymbol{o}_i - \boldsymbol{d}_i \parallel_2^2 = ... = \frac{1}{2n}\sum_{i=1}^{n}\sum_{j=0}^{m-1}(o_{ij} - d_{ij})^2$$

Είδαμε λοιπόν πως υπολογίζουμε το συνολικό σφάλμα του δικτύου χρησιμοποιώντας την συνάρτηση μέσου τετραγωνικού σφάλματος. Πως μπορούμε όμως βάσει αυτού του σφάλματος να ανανεώσουμε τις τιμές των συναπτικών βαρών; Παρατηρούμε ότι όσο μικρότερο το συνολικό σφάλμα του δικτύου, τόσο μεγαλύτερη η προβλεπτική του ικανότητα, καθώς τα διανύσματα εξόδου $\boldsymbol{o}_i$ θα ταυτίζονται όλο και περισσότερο με τα αντίστοιχα ιδανικά διανύσματα εξόδου $\boldsymbol{d}_i$. Συνεπώς, το πρόβλημα της εκπαίδευσης του δικτύου είναι στην ουσία του ισοδύναμο με το πρόβλημα της ελαχιστοποίησης του συνολικού σφάλματος.

Υπάρχουν σαφώς πολλοί διαφορετικοί αλγόριθμοι ελαχιστοποίησης συναρτήσεων. Ένας από τους πιο γνωστούς, ο οποίος βρίσκει εφαρμογή στην εκπαίδευση νευρωνικών δικτύων μέχρι και σήμερα, είναι ο αλγόριθμος «Κατάβασης Κλίσης» (Gradient Descent). Πρόκειται για έναν επαναληπτικό αλγόριθμο μέσω του οποίου σταδιακά οδηγούμαστε στην εύρεση ενός τοπικού ελαχίστου της συνάρτησης προς ελαχιστοποίηση. Ο αλγόριθμος έχει ως εξής: Έστω ότι καλούμαστε να ελαχιστοποιήσουμε μία συνάρτηση $f(\boldsymbol{x})$ ως προς ένα διάνυσμα μεταβλητών $\boldsymbol{x}$. Ξεκινώντας από ένα τυχαίο σημείο $\boldsymbol{x}^{(0)}$ ο αλγόριθμος πραγματοποιεί κάμποσες επαναλήψεις, σε κάθε μία από τις οποίες εκτελούνται τα δύο εξής βήματα:

1. Υπολόγισε την κλίση της συνάρτησης $f$ για το τωρινό σημείο $\boldsymbol{x}^{(t)}$.
2. Ανανέωσε το σημείο $\boldsymbol{x}^{(t)}$ ως εξής:

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \beta \cdot \nabla f(\boldsymbol{x}^{(t)})$$

Είναι γνωστό πως η αρνητική κλίση μίας συνάρτησης υπολογισμένη βάσει ενός οποιουδήποτε σημείου $\boldsymbol{x}$ έχει πάντοτε την κατεύθυνση της πιο απότομης μείωσης της συνάρτησης ως προς το σημείο αυτό. Συνεπώς, το μόνο που κάνει ο αλγόριθμος σε κάθε επανάληψη είναι να μετακινείται αργά και σταθερά προς την κατεύθυνση που του υποδεικνύει η κλίση, μειώνοντας έτσι σταδιακά την τιμή της συνάρτησης προς ελαχιστοποίηση. Ο όρος $\beta$ είναι ο ρυθμός μάθησης, του οποίου η λειτουργία αποτελεί την ρύθμιση του μεγέθους των «βημάτων» που εκτελεί ο αλγόριθμος, ακριβώς όπως γινόταν και στην περίπτωση του αλγορίθμου Perceptron. Χαμηλός ρυθμός μάθησης συνεπάγεται μικρά αλλά σταθερά βήματα, ενώ ένας υψηλός ρυθμός μάθησης, παρόλο που ενδέχεται να μας οδηγήσει σε ταχύτερη σύγκλιση, το πιο πιθανό είναι να μας βγάλει εκτός πορείας.

Επιστρέφοντας τώρα πίσω στα νευρωνικά δίκτυα, είναι προφανές ότι η έξοδος του δικτύου ουσιαστικά καθορίζεται από τα βάρη του. Αυτό σημαίνει ότι θα πρέπει να υπολογίσουμε την συνάρτηση σφάλματος ως προς τα συναπτικά βάρη, «μετακινώντας» τα έτσι σταδιακά ούτως ώστε το δίκτυο να είναι ικανό να παράγει έξοδο

χαμηλού σφάλματος. Βάσει του αλγορίθμου κατάβασης κλίσης, κάθε συναπτικό βάρος $w_{ij}$ θα ανανεώνεται ως εξής:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \beta \cdot \frac{\partial J}{\partial w_{ij}}$$

Με τον όρο $w_{ij}$ συμβολίζουμε το συναπτικό βάρος που συνδέει τους νευρώνες $i$ και $j$ του δικτύου ($i \to j$), ενώ ο όρος $\frac{\partial J}{\partial w_{ij}}$ αναφέρεται στην μερική παράγωγο της συνάρτησης σφάλματος ως προς το συναπτικό βάρος $w_{ij}$. Μέσω του κανόνα αλυσίδας μπορούμε να υπολογίσουμε την μερική παράγωγο του σφάλματος ως προς κάθε συναπτικό βάρος, καταλήγοντας έτσι σε έναν ενιαίο κανόνα ανανέωσής τον οποίο αποκαλούμε «Γενικευμένο Κανόνα Δέλτα» (Generalized Delta Rule)[2], και ο οποίος έχει την εξής μορφή:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \beta \cdot \delta_j \cdot x_i$$

Ο όρος $x_i$ αναφέρεται στην $i$-οστή τιμή του εκάστοτε διανύσματος εισόδου $\boldsymbol{x}$ του νευρώνα, η οποία προέρχεται από το αμέσως προηγούμενο επίπεδο του δικτύου, ή ακόμη και απευθείας από το επίπεδο εισόδου στην περίπτωση που ο νευρώνας $j$ ανήκει στο πρώτο κρυφό επίπεδο. Ο παραπάνω κανόνας μπορεί επίσης να εκφραστεί στην διανυσματική του μορφή ως εξής:

$$\boldsymbol{w}_j^{(t+1)} = \boldsymbol{w}_j^{(t)} + \beta \cdot \delta_j \cdot \boldsymbol{x}$$

όπου το $\boldsymbol{x}$ αποτελεί πλέον ολόκληρο το διάνυσμα εισόδου στον νευρώνα $j$ με συναπτικά βάρη $\boldsymbol{w}_j$.

Όσο αφορά τις τιμές δέλτα $\delta_j$, από τις οποίες και ο κανόνας παίρνει το όνομά του, αυτές υπολογίζονται βάσει ενός ελαφρώς διαφορετικού τρόπου αναλόγως του επιπέδου στο οποίο βρίσκεται ο εκάστοτε νευρώνας. Εάν ο νευρώνας $j$ αποτελεί μέρος του επιπέδου εξόδου του δικτύου, τότε η τιμή $\delta_j$ υπολογίζεται ως εξής:

$$\delta_j = \phi'(u_j) \cdot (d_j - o_j)$$

όπου με τον όρο $o_j$ συμβολίζουμε την έξοδο του νευρώνα $j$, ενώ ο όρος $d_j$ αντιστοιχεί στην $j$-οστή τιμή του εκάστοτε ιδανικού διανύσματος $\boldsymbol{d}$. Η συνάρτηση $\phi'$ δεν είναι τίποτε άλλο παρά μόνο η παράγωγος (ως προς την τιμή $u_j$) της συνάρτησης ενεργοποίησης $\phi$ του νευρώνα. Τέλος η τιμή $u_j$ αποτελεί μονάχα την έξοδο του νευρώνα $j$ πριν αυτή περάσει μέσα από την συνάρτηση ενεργοποίησης, δηλαδή ισχύει:

$$u_j = \langle \boldsymbol{x}, \boldsymbol{w}_j \rangle = w_{0j} + \sum_{i=1}^{l} x_i \cdot w_{ij}$$

όπου $l$ το πλήθος των νευρώνων του αμέσως προηγούμενου επιπέδου, δηλαδή το πλήθος των διαστάσεων του διανύσματος εισόδου $\boldsymbol{x}$ του νευρώνα $j$. Ως $w_{0j}$ συμβολίζετι ο όρος bias του νευρώνα.

[2] Πρόδρομος αυτού του κανόνα αποτελεί ο «Κανόνας Δέλτα» (Delta Rule) ο οποίος χρησιμοποιείται για την ανανέωση των συναπτικών βαρών Πολυστρωματικών Perceptron ενός κρυφού επιπέδου.

Εφόσον έχουμε καλύψει την περίπτωση ενός νευρώνα να ανήκει στο επίπεδο εξόδου, θα δούμε στην συνέχεια πως υπολογίζουμε την τιμή $\delta_j$ όταν ο νευρώνας $j$ ανήκει σε οποιοδήποτε κρυφό επίπεδο. Ο τύπος υπολογισμού είναι ο εξής:

$$\delta_j = \phi'(u_j) \cdot \sum_{i=0}^{r} w_{ji} \cdot \delta_i$$

Ο όρος $r$ που βρίσκεται εντός του παραπάνω τύπου αντιστοιχεί στο πλήθος των νευρώνων του αμέσως επόμενου επιπέδου. Συνεπώς παρατηρούμε ότι η τιμή $\delta_j$ του νευρώνα $j$ εξαρτάται από κάθε τιμή $\delta_i$ των νευρώνων του επόμενου επιπέδου.

Τώρα λοιπόν γίνεται κατανοητός ο λόγος για τον οποίο το σφάλμα μεταδίδεται από το τέλος του δικτύου προς την αρχή του. Είναι πλέον προφανές ότι όσο περισσότερο συμβάλλει ένας νευρώνας στο συνολικό σφάλμα του δικτύου, τόσο υψηλότερη θα είναι η απόλυτη τιμή του $\delta$. Στην περίπτωση των νευρώνων εξόδου, η τιμή $\delta$ εξαρτάται άμεσα από την διαφορά μεταξύ ιδανικής τιμής $d$ και πραγματικής εξόδου $o$. Συνεπώς εάν η τιμή εξόδου ενός νευρώνα απέχει αρκετά από την ιδανική έξοδο, τότε η τιμή του $|\delta|$ θα είναι αυξημένη. Στην περίπτωση τώρα που ένας νευρώνας $j$ ανήκει σε κάποιο κρυφό επίπεδο, τότε το μέγεθος $|\delta_j|$ θα είναι αυξημένο όταν τα επιμέρους γινόμενα $w_{ji} \cdot \delta_i$ είναι και αυτά αυξημένα, δηλαδή όταν αυτός συνδέεται μέσω υψηλών συναπτικών βάρών με νευρώνες $i$ του επόμενου επιπέδου, στους οποίους τυχαίνει να αντιστοιχούν υψηλές τιμές $|\delta_i|$. Το γεγονός αυτό μπορούμε να το ερμηνεύσουμε ως εξής: Ο νευρώνας $j$, λόγω των υψηλών συναπτικών βαρών του έχει μεγάλο «μερίδιο ευθύνης» για την έξοδο των νευρώνων $i$, και κατ΄ επέκταση για το συνολικό σφάλμα του δικτύου. Αυτό μπορούμε να το επαληθεύσουμε εάν μέσω του γενικευμένου κανόνα δέλτα εξάγουμε τον εξής τύπο μεταβολής των συναπτικών βαρών:

$$\Delta \boldsymbol{w}_j = -\beta \cdot \delta_j \cdot \boldsymbol{x}$$

Βάσει του παραπάνω τύπου παρατηρούμε ότι όσο μεγαλύτερο το μέγεθος της τιμής $|\delta_j|$, τόσο μεγαλύτερη θα είναι η μεταβολή που υφίστανται τα συναπτικά βάρη $\boldsymbol{w}_j$.

### 2.1.4 Τα Αναδρομικά Νευρωνικά Δίκτυα

Εφόσον έχουμε αναφερθεί στα νευρωνικά δίκτυα εμπρόσθιας διάδοσης στην συνέχεια θα εξετάσουμε τα «Αναδρομικά Νευρωνικά Δίκτυα» (Recurrent Neural Networks), τα οποία βασίζονται σε ένα συγκεκριμένο είδος αρχιτεκτονικής δικτύων, κατάλληλα προσαρμοσμένη για χρήση σε συνδυασμό με δεδομένα τα οποία παρουσιάζουν εξαρτήσεις μεταξύ τους, συνήθως μέσα στον χρόνο. Ένα απλό παράδειγμα αποτελεί ο ομιλούμενος/γραπτός λόγος, στα πλαίσια του οποίου κάθε λέξη εξαρτάται άμεσα από τις λέξεις που προέκυψαν πριν από αυτή. Ως προς την διαχείριση τέτοιου τύπου δεδομένων τα δίκτυα αυτά είναι εφοδιασμένα με ένα είδος «μνήμης» μέσω της οποίας διατηρείται η πληροφορία των προηγούμενων εισόδων. Αυτό σημαίνει ότι στην διαδικασία υπολογισμού της εξόδου ενός αναδρομικού δικτύου δεν συμβάλει μονάχα η τωρινή είσοδος, μα και οι υπόλοιπες είσοδοι οι οποίες έχουν τροφοδοτηθεί στο δίκτυο προηγουμένως.

Για να κατανοήσουμε την λειτουργία των αναδρομικών δικτύων θα αναλογιστούμε ένα απλό παράδειγμα. Έστω ότι έχουμε στην διάθεσή μας ένα σύνολο δεδομένων αποτελούμενο από ένα πλήθος άρθρων, τα οποία θέλουμε να ταξινομήσουμε σε κλάσεις. Τις κλάσεις αυτές θα μπορούσαν για παράδειγμα να αποτελούν διαφορεικά είδη άρθρου, δηλαδή επιστημονικό, πολιτικό, κίτρινος τύπος, κ.λπ. Κάθε άρθρο σαφώς απαρτίζεται από ένα σύνολο λέξεων τοποθετημένες σε μία συγκεκριμένη σειρά, τις οποίες έστω ότι έχουμε καταφέρει να αναπαραστίσουμε ως ξεχωριστά διανύσματα σε κάποιον χώρο $\mathbb{R}^d$. Εάν για παράδειγμα ένα αρθρο αποτελείται από $T$ λέξεις, τότε τα διανύσματα που αντιστοιχούν σε αυτό είναι τα $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T$, όπου το διάνυσμα $\boldsymbol{x}_1$ αντιστοιχεί στην πρώτη λέξη του άρθρου, το $\boldsymbol{x}_2$ στην δεύτερη και ούτω καθεξής.

Για λόγους απλότητας θα θεωρήσουμε ότι το αναδρομικό μας δίκτυό κατέχει ένα και μόνο κρυφό επίπεδο, το οποίο πράγματι πολλές φορές είναι επαρκές, ωστόσο δεν υπάρχει κάποιος περιορισμός σχετικά με το πλήθος των κρυφών επιπέδων. Σε αντίθεση με τα «κλασικά» νευρωνικά δίκτυα εμπρόσθιας διάδοσης, τα αναδρομικά δίκτυα διατηρούν ένα ξεχωριστό διάνυσμα $\boldsymbol{s}$ το οποίο αποκαλούμε «κατάσταση» του δικτύου. Το διάνυσμα αυτό αποτελεί την «μνήμη» του δικτύου, καθώς οι τιμές του εμπεριέχουν την πληροφορία που μας παρέχεται μέσω των προηγούμενων εισόδων. Τα συναπτικά βάρη τα οποία σχετίζονται με τον υπολογισμό του διανύσματος κατάστασης $\boldsymbol{s}$ διατηρούνται από το δίκτυο ξεχωριστά από τα υπόλοιπα. Με λίγα λόγια, μέσα σε ένα κρυφό επίπεδο ενός αναδρομικού δικτύου συναντάμε δύο διαφορετικά σύνολα νευρώνων:

1. Το πρώτο σύνολο νευρώνων, των οποίων τα συναπτικά βάρη μπορούμε να οργανώσουμε σε έναν πίνακα $\boldsymbol{U}$, δέχεται το διάνυσμα εισόδου $\boldsymbol{x}$ και παράγει ένα νέο διάνυσμα $\phi(\boldsymbol{U} \cdot \boldsymbol{x})$, το οποίο θα χρησιμοποιηθεί ως προς την ανανέωση του διανύσματος κατάστασης. Ως $\phi$ συμβολίζουμε την συνάρτηση ενεργοποίησης των νευρώνων.

2. Το δεύτερο σύνολο νευρώνων με συναπτικά βάρη $\boldsymbol{W}$ αφορά αποκλειστικά το διάνυσμα κατάστασης του δικτύου. Οι νευρώνες αυτοί δεν συνδέονται μόνο με το επόμενο επίπεδο του δικτύου, αλλά και μεταξύ τους, ούτως ώστε να παράγουν ένα νέο διάνυσμα κατάστασης $\boldsymbol{s'}$. Έχοντας υπολογίσει το διάνυσμα $\phi(\boldsymbol{W} \cdot \boldsymbol{s})$ μέσω αυτής της αναδρομικής σύνδεσης, το νέο διάνυσμα κατάστασης του δικτύου τελικά υπολογίζεται ως εξής:

   $$\boldsymbol{s'} = \phi(\boldsymbol{U} \cdot \boldsymbol{x}) + \phi(\boldsymbol{W} \cdot \boldsymbol{s})$$

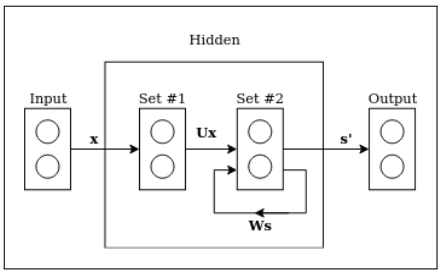
   Εφόσον η συνάρτηση $\phi$ εφαρμόζεται σε κάθε στοιχείο των διανυσμάτων $\boldsymbol{U} \cdot \boldsymbol{x}$ και $\boldsymbol{W} \cdot \boldsymbol{s}$ ξεχωριστά, η παραπάνω σχέση μπορεί να εκφραστεί ισοδύναμα ως:

   $$\boldsymbol{s'} = \phi(\boldsymbol{U} \cdot \boldsymbol{x} + \boldsymbol{W} \cdot \boldsymbol{s})$$

Τέλος, το ανανεωμένο διάνυσμα κατάστασης $\boldsymbol{s'}$ τροφοδοτείται στο επίπεδο εξόδου του δικτύου με συναπτικά βάρη $\boldsymbol{V}$ έτσι ώστε να υπολογίσουμε το τελικό διάνυσμα εξόδου $\boldsymbol{o} = \boldsymbol{V} \cdot \boldsymbol{s'}$.

Επιστρέφοντας πίσω στο παράδειγμά μας, έστω ότι στην είσοδο του δικτύου καταφτάνει ένα διάνυσμα $\boldsymbol{x}_1$ το οποίο αναπαριστά την πρώτη λέξη ενός άρθρου. Εφόσον το δίκτυο δεν έχει συναντήσει κάποιο άλλο διάνυσμα έως τώρα, η αρχική του κατάσταση $\boldsymbol{s}_0$ αρχικοποιείται από εμάς, συνήθως ως το μηδενικό διάνυσμα. Βάσει των διανυσμάτων $\boldsymbol{x}_1$ και $\boldsymbol{s}_0$ υπολογίζεται το νέο διάνυσμα κατάστασης $\boldsymbol{s}_1$ του δικτύου:

Σχήμα 2.4: Η λειτουργία του κρυφού επιπέδου ενός αναδρομικού δικτύου.




$$\boldsymbol{s}_1 = \phi(\boldsymbol{U} \cdot \boldsymbol{x}_1 + \boldsymbol{W} \cdot \boldsymbol{s}_0)$$

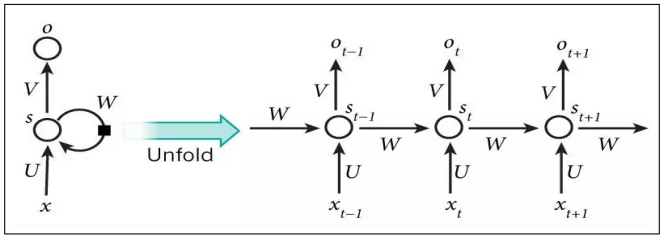
Σε αυτό το σημείο μπορούμε βέβαια να υπολογίσουμε την έξοδο $\boldsymbol{o}_1$, η οποία αν και σαφώς ενδέχεται να έχει κάποια χρησιμότητα σε άλλου είδους προβλήματα, στο παράδειγμά μας το μόνο που μας ενδιαφέρει είναι η τελική έξοδος $\boldsymbol{o}_T$ του δικτύου, καθώς την στιγμή $T$ το αντίστοιχο διάνυσμα κατάστασης $\boldsymbol{s}_T$ θα περιέχει πλέον την πληροφορία κάθε προηγούμενης εισόδου, δηλαδή κάθε λέξης του εκάστοτε άρθρου. Συνεπώς, μπορούμε να παραλείψουμε τον υπολογισμό της εξόδου $\boldsymbol{o}_1$ και να προχωρήσουμε απευθείας στον υπολογισμό της νέας κατάστασης $\boldsymbol{s}_2$ βάσει της τωρινής εισόδου $\boldsymbol{x}_2$ και της προηγούμενης κατάστασης $\boldsymbol{s}_1$:

$$\boldsymbol{s}_2 = \phi(\boldsymbol{U} \cdot \boldsymbol{x}_2 + \boldsymbol{W} \cdot \boldsymbol{s}_1)$$

Βλέπουμε λοιπόν, ότι ο γενικός κανόνας ενημέρωσης του διανύσματος κατάστασης δίνεται από τον εξής τύπο:

$$\boldsymbol{s}_t = \phi(\boldsymbol{U} \cdot \boldsymbol{x}_t + \boldsymbol{W} \cdot \boldsymbol{s}_{t-1}) \ , \ \forall t = 1, ..., T$$

Σχήμα 2.5: Ένα αναδρομικό νευρωνικό δίκτυο ενός κρυφού επιπέδου «ξεδιπλωμένο» στον χρόνο.




Πηγή: `https://arxiv.org/ftp/arxiv/papers/1902/1902.07250.pdf`

Η παραπάνω διαδικασία επαναλαμβάνεται μέχρι να υπολογίσουμε και το διάνυσμα κατάστασης $\boldsymbol{s}_T$, βάσει του οποίου τελικά υπολογίζεται η έξοδος $\boldsymbol{o}_T = \boldsymbol{V} \cdot \boldsymbol{s}_T$, η οποία αποτελεί και την τελική πρόβλεψη για την κλάση του άρθρου. Όταν ένα

νέο άρθρο φτάσει στη είσοδο, τότε θα πρέπει να επαναρχικοποιήσουμε το διάνυσμα κατάστασης του δικτύου, «ξεχνώντας» με αυτόν τον τρόπο όλες τις προηγούμενες εισόδους που σχετίζονταν με το προηγούμενο άρθρο, και η ίδια διαδικασία επαναλαμβάνεται εκ νέου.

Θα πρέπει και πάλι να τονίσουμε ότι ένα αναδρομικό νευρωνικό δίκτυο ενδέχεται να αποτελείται από κάμποσα κρυφά επίπεδα. Σε αυτήν την περίπτωση η διαδικασία που ακολουθείται παραμένει η ίδια με την μοναδική διαφορά να συνίσταται στο γεγονός ότι το δίκτυο πλέον περιέχει περισσότερα του ενός διανύσματα κατάστασης. Κάθε κρυφό επίπεδο διατηρεί το δικό του διάνυσμα κατάστασης, το οποίο ανανεώνεται βάσει της μεθόδου που περιγράψαμε παραπάνω έτσι ώστε στην συνέχεια να τροφοδοτηθεί ως είσοδο στο αμέσως επόμενο επίπεδο. Η διαδικασία αυτή επαναλαμβάνεται έως ότου να λάβουμε την έξοδο του δικτύου. Συνεπώς γίνεται κατανοητό ότι τα αναδρομικά δίκτυα διατηρούν την ίδια «μη-αναδρομικότητα» με τα κλασικά δίκτυα όσο αφορά την οργάνωση των επιπέδων τους, ενώ το στοιχείο της «αναδρομικότητας» παρατηρείται μονάχα εντός των κρυφών επιπέδων.

### 2.1.5 Ο Αλγόριθμος Backpropagation Through Time

Έχοντας μιλήσει για τον εσωτερικό μηχανισμό των αναδρομικών δικτύων και πως αυτά παράγουν την τελική έξοδο, στην συνέχεια θα δούμε πως πραγματοποιείται η εκπαίδευσή τους. Θα προσπαθήσουμε να εφαρμόσουμε τον αλγόριθμο Backpropagation στο δίκτυο του προηγούμενου παραδείγματος, στα πλαίσια του οποίου το σύνολο δεδομένων αποτελείται από ένα πλήθος άρθρων της μορφής $(\{\boldsymbol{x_t}\}_{t=1}^{T}, y)$. Αυτή η σχέση εισόδου-εξόδου αποτελεί μία σχέση τύπου «πολλά-προς-ένα», καθώς σε κάθε ακολουθία διανυσμάτων $\{\boldsymbol{x_t}\}_{t=1}^{T}$ αντιστοιχεί μία και μόνο έξοδος $y$ η οποία εκφράζει την κλάση της, δηλαδή την κατηγορία του άρθρου. Υπενθυμίζουμε ότι οι δύο βασικές εξισώσεις του δικτύου, αυτές της ανανέωσης του διανύσματος κατάστασης και του υπολογισμού της εξόδου, είναι οι εξής:

$$\boldsymbol{s}_t = \phi(\boldsymbol{U} \cdot \boldsymbol{x}_t + \boldsymbol{W} \cdot \boldsymbol{s}_{t-1}) \quad \text{και} \quad \boldsymbol{o}_t = \boldsymbol{V} \cdot \boldsymbol{s}_t$$

Εφόσον το σφάλμα του δικτύου εξαρτάται μόνο από την τελική εξόδο $\boldsymbol{o}_T$, μπορούμε βάσει αυτού να υπολογίσουμε τις κλίσεις των συναπτικών βαρών έτσι ώστε να ανανεώσουμε τις τιμές τους. Έχοντας υπολογίσει το σφάλμα $J_T$ βάσει του διανύσματος εξόδου $\boldsymbol{o}_T$, παραγωγίζουμε ως προς τα συναπτικά βάρη $\boldsymbol{V}$ του επιπέδου εξόδου για να λάβουμε τον εξής τύπο:

$$\frac{\partial J_T}{\partial \boldsymbol{V}} = \frac{\partial J_T}{\partial \boldsymbol{o_T}} \cdot \frac{\partial \boldsymbol{o}_T}{\partial \boldsymbol{V}}$$

Ο υπολογισμός των παραπάνω τιμών αποδεικνύεται αρκετά απλός. Όσο αφορά τα συναπτικά βάρη $\boldsymbol{W}$ και $\boldsymbol{U}$, παραγωγίζοντας το σφάλμα λαμβάνουμε:

$$\frac{\partial J_T}{\partial \boldsymbol{W}} = \frac{\partial J_T}{\partial \boldsymbol{o_T}} \cdot \frac{\partial \boldsymbol{o}_T}{\partial \boldsymbol{s}_T} \cdot \frac{\partial \boldsymbol{s}_T}{\partial \boldsymbol{W}} \quad \text{και} \quad \frac{\partial J_T}{\partial \boldsymbol{U}} = \frac{\partial J_T}{\partial \boldsymbol{o_T}} \cdot \frac{\partial \boldsymbol{o}_T}{\partial \boldsymbol{s}_T} \cdot \frac{\partial \boldsymbol{s}_T}{\partial \boldsymbol{U}}$$

Το πρόβλημα που προκύπτει σε αυτήν την περίπτωση οφείλεται στο γεγονός ότι το τελικό διάνυσμα κατάστασης $\boldsymbol{s}_T$ εξαρτάται από όλα τα προηγούμενα διανύσματα

κατάστασης $\{\boldsymbol{s}_t\}_{t=0}^{T-1}$, με αποτέλεσμα ο υπολογισμός των δύο κλίσεων να είναι αρκετά πιο περίπλοκος σε σχέση με τον προηγούμενο. Εφόσον κάθε προηγούμενη κατάσταση έχει συμβάλει στον υπολογισμό της τελικής κατάστασης $\boldsymbol{s}_T$, θα πρέπει κάθε ένα από αυτά τα διανύσματα να πάρει μέρος στον υπολογισμό της κλίσης. Συνεπώς καταλήγουμε στους παρακάτω δύο τύπους:

$$\frac{\partial J_T}{\partial \boldsymbol{W}} = \sum_{k=0}^{T} \frac{\partial J_T}{\partial \boldsymbol{o_T}} \cdot \frac{\partial \boldsymbol{o}_T}{\partial \boldsymbol{s}_T} \cdot \frac{\partial \boldsymbol{s}_T}{\partial \boldsymbol{s}_k} \cdot \frac{\partial \boldsymbol{s}_k}{\partial \boldsymbol{W}} \quad \text{και} \quad \frac{\partial J_T}{\partial \boldsymbol{U}} = \sum_{k=0}^{T} \frac{\partial J_T}{\partial \boldsymbol{o_T}} \cdot \frac{\partial \boldsymbol{o}_T}{\partial \boldsymbol{s}_T} \cdot \frac{\partial \boldsymbol{s}_T}{\partial \boldsymbol{s}_k} \cdot \frac{\partial \boldsymbol{s}_k}{\partial \boldsymbol{U}}$$

Θα πρέπει να τονίσουμε ότι κάθε παράγωγος $\frac{\partial \boldsymbol{s}_T}{\partial \boldsymbol{s}_k}$, για $k \in \{1, ..., T\}$ θα πρέπει επίσης να υπολογιστεί μέσω του κανόνα αλυσίδας, καθώς κάθε διάνυσμα κατάστασης $\boldsymbol{s}_k$ εξαρτάται από όλες τις προηγούμενες καταστάσεις. Συνεπώς, καταλήγουμε στους τρεις τελικούς τύπους για τον υπολογισμό της κλίσης των συναπτικών βαρών:

- $\dfrac{\partial J_T}{\partial \boldsymbol{V}} = \dfrac{\partial J_T}{\partial \boldsymbol{o_T}} \cdot \dfrac{\partial \boldsymbol{o}_T}{\partial \boldsymbol{V}}$
- $\dfrac{\partial J_T}{\partial \boldsymbol{W}} = \displaystyle\sum_{k=0}^{T} \frac{\partial J_T}{\partial \boldsymbol{o_T}} \cdot \frac{\partial \boldsymbol{o}_T}{\partial \boldsymbol{s}_T} \cdot \left( \prod_{j=k+1}^{T} \frac{\partial \boldsymbol{s}_j}{\partial \boldsymbol{s}_{j-1}} \right) \cdot \frac{\partial \boldsymbol{s}_k}{\partial \boldsymbol{W}}$
- $\dfrac{\partial J_T}{\partial \boldsymbol{U}} = \displaystyle\sum_{k=0}^{T} \frac{\partial J_T}{\partial \boldsymbol{o_T}} \cdot \frac{\partial \boldsymbol{o}_T}{\partial \boldsymbol{s}_T} \cdot \left( \prod_{j=k+1}^{T} \frac{\partial \boldsymbol{s}_j}{\partial \boldsymbol{s}_{j-1}} \right) \cdot \frac{\partial \boldsymbol{s}_k}{\partial \boldsymbol{U}}$

Η ενημέρωση των συναπτικών βαρών βάσει των παραπάνω τριών τύπων αποτελεί μία παραλλαγή του αλγορίθμου Backpropagation γνωστή ως «Backpropagation Through Time» (BPTT), καθώς αυτό που συμβαίνει είναι στην ουσία μονάχα η εφαρμογή του κανονικού αλγορίθμου Backpropagation μέσα στον «χρόνο», συμπεριλαμβάνοντας δηλαδή κάθε διάνυσμα καταστάσης $\boldsymbol{s}_t$. Είδαμε λοιπόν τι γίνεται όταν το σύνολο δεδομένων πάνω στο οποίο εργαζόμαστε αποτελείται από στοιχεία της σχέσης «πολλά-προς-ένα». Τι συμβαίνει όμως στην περίπτωση που η σχέση εισόδου-εξόδου είναι τύπου «πολλά-προς-πολλά», δηλαδή όταν σε κάθε ακολουθία εισόδου $\{\boldsymbol{x_t}\}_{t=1}^{T}$ αντιστοιχεί μία ακολουθία εξόδου $\{y_t\}_{t=1}^{T}$;

Έστω ότι θέλουμε να εκπαιδεύσουμε ένα αναδρομικό δίκτυο το οποίο θα μπορεί να προβλέπει την γραμματική σημασία των λέξεων μίας πρότασης, δηλαδή εάν μία λέξη είναι ουσιαστικό, ρήμα, επίθετο, κ.λπ. Συνεπώς μία πρόταση αναπαριστάνεται ως μία ακολουθία λέξεων, δηλαδή μία ακολουθία διανυσμάτων $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T$. Σε αυτό το πρόβλημα παρατηρούμε ότι δεν μας ενδιαφέρει μονάχα η τελική έξοδος $\boldsymbol{o}_T$ της ακολουθίας, μα ολόκληρη η ακολουθία εξόδου $\{\boldsymbol{o_t}\}_{t=1}^{T}$, καθώς κάθε διάνυσμα εξόδου $\boldsymbol{o}_t$ θα αποτελεί την πρόβλεψη του δικτύου σχετικά με την γραμματική σημασία της $t$-οστής λέξης της εκάστοτε πρότασης. Σε αυτήν την περίπτωση, το συνολικό σφάλμα μίας ακολουθίας υπολογίζεται ως η μέση τιμή των επιμέρους σφαλμάτων που προκύπτουν από κάθε στοιχείο της ακολουθίας, δηλαδή θα ισχύει:

$$J = \frac{1}{T} \sum_{t=1}^{T} J_t$$

Κάθε τιμή $J_t$ υπολογίζεται βάσει του εκάστοτε διανύσμτος εξόδου $\boldsymbol{o}_t$. Αντιστοίχως παραγωγίζοντας το συνολικό σφάλμα ως προς τα συναπτικά βάρη $\boldsymbol{V}$, $\boldsymbol{W}$ και $\boldsymbol{U}$, παρατηρούμε ότι η παράγωγος που προκύπτει επίσης εξαρτάται από όλες τις επιμέρους κλίσεις:

$$\frac{\partial J}{\partial \boldsymbol{\Theta}} = \frac{\partial}{\partial \boldsymbol{\Theta}} \left( \frac{1}{T} \sum_{t=1}^{T} J_t \right) = \frac{1}{T} \sum_{t=1}^{T} \frac{\partial J_t}{\partial \boldsymbol{\Theta}} \ , \quad \text{όπου} \ \ \boldsymbol{\Theta} \in \{\boldsymbol{V}, \boldsymbol{W}, \boldsymbol{U}\}$$

Συνεπώς σε αυτήν την περίπτωση οι τύποι υπολογισμού των κλίσεων ως προς τα τρία είδη συναπτικών βαρών ορίζονται ως εξής:

- $\dfrac{\partial J}{\partial \boldsymbol{V}} = \dfrac{1}{T} \displaystyle\sum_{t=1}^{T} \frac{\partial J_t}{\partial \boldsymbol{o_t}} \cdot \frac{\partial \boldsymbol{o}_t}{\partial \boldsymbol{V}}$
- $\dfrac{\partial J}{\partial \boldsymbol{W}} = \dfrac{1}{T} \displaystyle\sum_{t=1}^{T} \left[ \sum_{k=0}^{t} \frac{\partial J_t}{\partial \boldsymbol{o_t}} \cdot \frac{\partial \boldsymbol{o}_t}{\partial \boldsymbol{s}_t} \cdot \left( \prod_{j=k+1}^{t} \frac{\partial \boldsymbol{s}_j}{\partial \boldsymbol{s}_{j-1}} \right) \cdot \frac{\partial \boldsymbol{s}_k}{\partial \boldsymbol{W}} \right]$
- $\dfrac{\partial J}{\partial \boldsymbol{U}} = \dfrac{1}{T} \displaystyle\sum_{t=1}^{T} \left[ \sum_{k=0}^{t} \frac{\partial J_t}{\partial \boldsymbol{o_t}} \cdot \frac{\partial \boldsymbol{o}_t}{\partial \boldsymbol{s}_t} \cdot \left( \prod_{j=k+1}^{t} \frac{\partial \boldsymbol{s}_j}{\partial \boldsymbol{s}_{j-1}} \right) \cdot \frac{\partial \boldsymbol{s}_k}{\partial \boldsymbol{U}} \right]$

Θα πρέπει να σημειώσουμε ότι οι τύποι που έχουμε εξάγει έως τώρα ισχύουν μονάχα όταν το αναδρομικό δίκτυο περιέχει ένα και μοναδικό κρυφό επίπεδο. Στην περίπτωση ενός δικτύου πολλαπλών κρυφών επιπέδων οι υπολογισμοί προφανώς περιπλέκονται ακόμη περισσότερο, καθώς κάθε επίπεδο διατηρεί το δικό του διάνυσμα κατάστασης, το οποίο ανά χρονική στιγμή εξαρτάται τόσο από το ίδιο κατά τις προηγούμενες χρονικές στιγμές, όσο και από τα διανύσματα καταστάσεων των προηγούμενων κρυφών επιπέδων. Ήδη στην περίπτωση χρήσης ενός κρυφού επιπέδου, βλέπουμε ότι οι υπολογισμοί των κλίσεων αποτελούν γινόμενο πολλών όρων, των οποίων οι τιμές ενδέχεται να είναι μικρότερες της μονάδας. Αυτό με την σειρά του μπορεί εύκολα να οδηγήσει στο πρόβλημα της «Εξαφανιζόμενης Κλίσης» (Vanishing Gradient), καθιστώντας την εκπαίδευση των αναδρομικών δικτύων μία αρκετά δύσκολη υπόθεση [3]. Επιπλέον, παρόλο που τα αναδρομικά δίκτυα είναι στην θεωρία ικανά να μάθουν εξαρτήσεις μακράς διαρκείας, στην πράξη η πληροφορία των προηγούμενων εισόδων χάνεται αρκετά γρήγορα. Λόγω των παραπάνω η έρευνα σταδιακά στράφηκε προς την κατασκευή ακόμη πιο ισχυρών μοντέλων τα οποία θα μπορούσαν να αντιμετωπίσουν τέτοιου είδους προβλήματα.

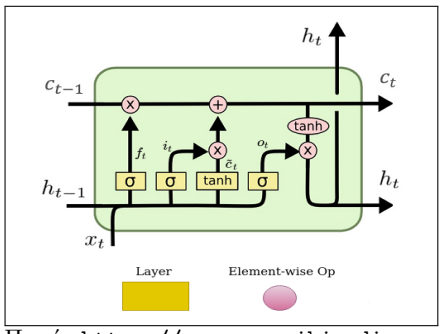
### 2.1.6 Τα Δίκτυα Long Short-Term Memory

Τα δίκτυα Long Short-Term Memory (LSTM) [4] αποτελούν κατά κάποιον τρόπο την εξέλιξη των αναδρομικών νευρωνικών δικτύων. Όπως και οι πρόδρομοί τους έτσι και τα δίκτυα LSTM εφαρμόζονται κυρίως στην επίλυση προβλημάτων όπου τα δεδομένα παρουσιάζουν εξαρτήσεις μεταξύ τους. Μία μονάδα LSTM ορίζεται στην ουσία ως μία συγκεκριμένη ακολουθία υπολογισμών, οι οποίες βασίζονται σε

ένα πλήθος διανυσμάτων και συναπτικών βαρών. Ένα LSTM δίκτυο μπορεί να αποτελείται από μία και μοναδική LSTM μονάδα, η ακόμη και από περισσότερες οι οποίες είναι σειριακά συνδεδεμένες μεταξύ τους, ακριβώς όπως και τα κρυφά επίπεδα ενός κλασικού δικτύου.

Για να καταλάβουμε τον εσωτερικό μηχανισμό μίας LSTM μονάδας θα πρέπει να εισάγουμε πέντε έννοιες, στην καθεμία από τις οποίες αντιστοιχεί και από ένα διαφορετικό διάνυσμα το οποίο συμμετέχει στους υπολογισμούς που λαμβάνουν μέρος εντός της μονάδας. Αυτές είναι η «θύρα λησμονιάς» καθώς και οι «θύρες εισόδου» και «εξόδου» (forget, input and output gates) στις οποίες αντιστοιχούν τα διανύσματα $\boldsymbol{f}_t$, $\boldsymbol{i}_t$ και $\boldsymbol{o}_t$ αντίστοιχα, η «κατάσταση κυττάρου» (cell state) $\boldsymbol{c}_t$, και τέλος η «κρυφή κατάσταση» (hidden state) $\boldsymbol{h}_t$ της μονάδας, η οποία αποτελεί και την βασική της έξοδό. Η διαδικασία υπολογισμού της εξόδου έχει ως εξής: Δεδομένου ενός νέου διανύσματος εισόδου $\boldsymbol{x}_t$ μίας ακολουθίας $\{\boldsymbol{x}_t\}_{t=1}^{T}$, καθώς και των δύο διανυσμάτων κατάστασης $\boldsymbol{c}_{t-1}$ και $\boldsymbol{h}_{t-1}$, τα οποία εκφράζουν την κατάσταση της μονάδας την στιγμή που το διάνυσμα $\boldsymbol{x}_t$ φτάνει στην είσοδο, τα βήματα που εκτελούνται ως προς την ανανέωση των καταστάσεων, δηλαδή τον υπολογισμό των διανυσμάτων $\boldsymbol{c}_t$ και $\boldsymbol{h}_t$, είναι τα εξής:

Σχήμα 2.6: Μία μονάδα LSTM.




Πηγή: `https://commons.wikimedia.org/wiki/File:LSTM.png`

1. Η μονάδα αρχικά τροφοδοτεί τα διανύσματα $\boldsymbol{x}_t$ και $\boldsymbol{h}_{t-1}$ στην θύρα λησμονιάς με σκοπό τον υπολογισμό του διανύσματος $\boldsymbol{f}_t$:

$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}_f \cdot (\boldsymbol{x}_t \oplus \boldsymbol{h}_{t-1})\right)$$

   Με το σύμβολο «$\oplus$» ορίζουμε την πράξη της «συνένωσης» (concatenation) δύο διανυσμάτων, ενώ η συνάρτηση $\sigma : \mathbb{R} \to [0, 1]$ αντοιστοιχεί στην σιγμοειδή συνάρτηση, η οποία εφαρμόζεται σε κάθε στοιχείο του διανύσματος που προκύπτει από την πράξη $\boldsymbol{W}_f \cdot (\boldsymbol{x}_t \oplus \boldsymbol{h}_{t-1})$ έτσι ώστε να λάβουμε το διάνυσμα $\boldsymbol{f}_t$. Συνεπώς, η θύρα λησμονιάς δεν είναι τίποτε άλλο παρά μόνο ένα απλό επίπεδο σιγμοειδούς συνάρτησης ενεργοποίησης, το οποίο δέχεται ως είσοδο την συνένωση των διανυσμάτων $\boldsymbol{x}_t$ και $\boldsymbol{h}_{t-1}$. Θα μπορούσαμε να πούμε ότι αυτή η θύρα κατά κάποιο τρόπο επιλέγει ποιό μέρος της υπάρχουσας πληροφορίας θα «ξεχαστεί» από την κατάσταση κυττάρου, καθώς όπως θα δούμε στην συνέχεια το διάνυσμα $\boldsymbol{f}_t$ πρόκειται να πολλαπλασιαστεί κατά Hadamard (δηλαδή πολλαπλασιασμός στοιχείου με στοιχείο) με το διάνυσμα κατάστασης κυττάρου $\boldsymbol{c}_{t-1}$. Καθώς οι τιμές του διανύσματος $\boldsymbol{f}_t$ θα ανήκουν στο διάστημα $[0, 1]$, μέσω του πολλαπλασιασμού Hadamard μερικές τιμές θα «ξεχαστούν» από το διάνυσμα $\boldsymbol{c}_{t-1}$ (πολλαπλασιασμός με τιμές κοντά στο μηδέν) και άλλες θα παραμείνουν ως έχει (πολλαπλασιασμός με τιμές κοντά στην μονάδα).

2. Εφόσον έχουμε εξακριβώσει ποιά πληροφορία θα «ξεχαστεί», στην συνέχεια υπολογίζεται η καινούρια πληροφορία που πρόκειται να ενσωματωθεί στην κατάσταση κυττάρου. Η διαδικασία αυτή περιλαμβάνει τρία επιμέρους βήματα:

   (α′) Το διάνυσμα $\boldsymbol{x}_t \oplus \boldsymbol{h}_{t-1}$ αρχικά τροφοδοτείται σε ένα επίπεδο με συνάρτηση ενεργοποίησης την υπερβολική εφαπτομένη $\tanh : \mathbb{R} \to [-1, 1]$, έτσι ώστε να λάβουμε το διάνυσμα υποψήφιων τιμών $\tilde{\boldsymbol{c}}$ βάσει των οποίων πρόκειται να ανανεωθεί η κατάσταση κυττάρου:

   $$\tilde{\boldsymbol{c}} = \tanh\left(\boldsymbol{W}_{\tilde{\boldsymbol{c}}} \cdot (\boldsymbol{x}_t \oplus \boldsymbol{h}_{t-1})\right)$$

   (β′) Το επόμενο βήμα συνίσταται στον υπολογισμό του διανύσματος $\boldsymbol{i}_t$ της θύρας εισόδου. Όπως και η θύρα λησμονιάς, έτσι και η θύρα εισόδου αποτελεί μονάχα ένα επίπεδο σιγομοειδούς συνάρτησης ενεργοποίησης, το οποίο δέχεται την ίδια ακριβώς είσοδο:

   $$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}_i \cdot (\boldsymbol{x}_t \oplus \boldsymbol{h}_{t-1})\right)$$

   (γ′) Τέλος πραματοποιείται πολλαπλασιασμός Hadamard μεταξύ των διανυσμάτων $\tilde{\boldsymbol{c}}$ και $\boldsymbol{i}_t$. Μέσω της πράξης αυτής το διάνυσμα $\boldsymbol{i}_t$ επιδρά πάνω στο διάνυσμα $\tilde{\boldsymbol{c}}$, αποφασίζοντας κατά αυτόν τον τρόπο ποιες τιμές του «αξίζει» να διατηρηθούν και ποιές όχι:

   $$\boldsymbol{u}_t = \tilde{\boldsymbol{c}} \circ \boldsymbol{i}_t$$

   Το τελικό διάνυσμα $\boldsymbol{u}_t$ περιέχει τις τιμές οι οποίες πρόκειται να χρησιμοποιηθούν ως προς την ανανέωση της κατάστασης κυττάρου.

3. Σειρά έχει η ανανέωση της κατάστασης κυττάρου:

   $$\boldsymbol{c}_t = \boldsymbol{f}_t \circ \boldsymbol{c}_{t-1} + \boldsymbol{u}_t$$

   Το διάνυσμα $\boldsymbol{c}_{t-1}$ πολλαπλασιάζεται κατά Hadamard με το διάνυσμα $\boldsymbol{f}_t$ με σκοπό να «ξεχάσει» μέρος της προηγούμενης πληροφορίας, ενώ το διάνυσμα $\boldsymbol{u}_t$ προστίθεται στο αποτέλεσμα της πράξης αυτής, αποδίδοντας έτσι την όποια νέα πληροφορία.

4. Τέλος θα πρέπει να ανανεωθεί και το διάνυσμα κρυφής κατάστασης $\boldsymbol{h}_{t-1}$ της μονάδας, το οποίο θα αποτελέσει και την τελική έξοδο. Αρχικά το διάνυσμα $\boldsymbol{x}_t \oplus \boldsymbol{h}_{t-1}$ τροφοδοτείται στην θύρα εξόδου, η οποία επίσης αποτελεί ένα απλό επίπεδο σιγμοειδούς συνάρτησης ενεργοποίησης:

   $$\boldsymbol{o}_t = \sigma\left(\boldsymbol{W}_o \cdot (\boldsymbol{x}_t \oplus \boldsymbol{h}_{t-1})\right)$$

   Στην συνέχεια το διάνυσμα κρυφής κατάστασης ανανεώνεται εκτελώντας πολλαπλασιασμό Hadamard μεταξύ του διανύσματος $\boldsymbol{o}_t$ και του ανανεωμένου διανύσματος κατάστασης κυττάρου $\boldsymbol{c}_t$, αφού πρώτα οι τιμές του περάσουν μέσα από την συνάρτηση υπερβολικής εφαπτομένης:

   $$\boldsymbol{h}_t = \boldsymbol{o}_t \circ \tanh(\boldsymbol{c}_t)$$

   Το διάνυσμα $\boldsymbol{h}_t$ αποτελεί πλέον το διάνυσμα εξόδου ολόκληρης της μονάδας.

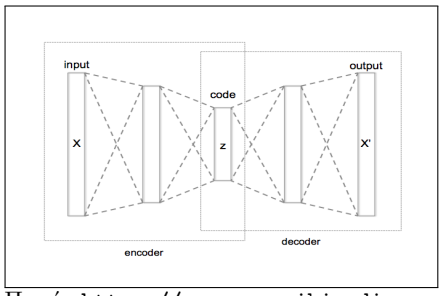
Συνεπώς κάθε LSTM μονάδα αποτελείται από έναν συνδυασμό τεσσάρων «επιπέδων» και πράξεων μεταξύ διανυσμάτων, τα οποία εκτελούνται με μία αυστηρά καθορισμένη σειρά. Μέσω αυτής της διαδικασίας το δίκτυο είναι ικανό να μάθει περίπλοκες εξαρτήσεις μεταξύ των δεδομένων, κάνοντας χρήση ακόμη και μίας μόνο μονάδας. Για το λόγο αυτό, τα δίκτυα LSTM συνεχίζουν έως και σήμερα να αποτελούν μία από τις βασικές επιλογές μοντέλων όσο αφορά την επίλυση τέτοιου είδους προβλημάτων.

### 2.1.7 Οι Αυτοκωδικοποιητές

Η τελευταία κατηγορία τεχνητών νευρωνικών δικτύων που θα εξετάσουμε αφορά τους «Αυτοκωδικοποιητές» (Autoencoders), ένα είδος δικτύου το οποίο διαφέρει σε μεγάλο βαθμό από τα μοντέλα που έχουμε δει έως τώρα, και το οποίο αποτελεί προϊόν μακρόχρονης και συλλογικής έρευνας με την πρώτη αναφορά σε αυτό να εντοπίζεται την δεκαετία του 80′ [5]. Αρχικά θα πρέπει να τονίσουμε ότι τα δίκτυα αυτά δεν χρησιμοποιούνται για την αντιμετώπιση προβλημάτων ταξινόμησης, αλλά η λειτουργία τους αποβλέπει κυρίως στην επίλυση προβλημάτων που σχετίζονται με τα ίδια τα δεδομένα, όπως για παράδειγμα είναι η Επιλογή Χαρακτηριστικών (Feature Selection), η Συμπίεση (Compression), η Ελάττωση Θορύβου (Noise Reduction) κ.λπ. Αυτό σημαίνει ότι οι Αυτοκωδικοποιητές στην ουσία αποτελούν μοντέλα Μη-Επιβλεπόμενης Μάθησης, δηλαδή η εκπαίδευσή τους δεν χρήζει των ετικετών των δεδομένων.

Αρχικά θα πρέπει να αναφερθούμε στην αρχιτεκτονική η οποία διέπει τα μοντέλα αυτά. Ένας κλασσικός Αυτοκωδικοποιητής είναι στην ουσία ένα νευρωνικό δίκτυο εμπρόσθιας διάδοσης[3], τα κρυφά επίπεδα του οποίου κατασκευάζονται ακολουθώντας μία συγκεκριμένη λογική. Στηριζόμενοι στην αρχιτεκτονική των Αυτοκωδικοποιητών μπορούμε να διαχωρίσουμε τα στοιχεία τα οποία απαρτίζουν ένα τέτοιο δίκτυο σε τρία ξεχωριστά μέρη, τα οποία αποκαλούμε «Κωδικοποιητή» (Encoder), επίπεδο «Bottleneck» και «Αποκωδικοποιητή» (Decoder) του δικτύου αντίστοιχα. Για να κατανοήσουμε καλύτερα την λειτουργία των Αυτοκωδικοποιητών θα αναλογιστούμε ότι καλούμαστε να επιλύσουμε το πρόβλημα της Επιλογής Χαρακτηριστικών. Θα θέλαμε δηλαδή δεδομένου ένος συνόλου δεδομένων να εξάγουμε ένα σύνολο χαρακτηριστικών για κάθε στοιχείο, το οποίο αν και μικρότερο από το αρχικό, μπορεί να περιγράψει τα δεδομένα χωρίς να χαθεί σημαντική πληροφορία. Πρώτα όμως θα αναφερθούμε λεπτομερέστερα

Σχήμα 2.7: Η Αρχιτεκτονική ενός Αυτοκωδικοποιητή.




Πηγή: `https://commons.wikimedia.org/wiki/File:Autoencoder\_structure.png`

[3]Παρόλα αυτά η χρήση διαφόρων ειδών επιπέδων όπως RNN ή LSTM μονάδων είναι επίσης εφικτή.

στα τρία μέρη που απαρτίζουν έναν Αυτοκωδικοποιητή:

- Κωδικοποιητής: Το τμήμα αυτό του δικτύου περιέχει το επίπεδο εισόδου καθώς και ένα πλήθος κρυφών επιπέδων, το μέγεθος των οποίων μειώνεται σταδιακά. Για παράδειγμα εάν τα στοιχεία $\boldsymbol{x}$ του συνόλου δεδομένων που έχουμε στην διάθεσή μας ανήκουν στον χώρο $\mathbb{R}^d$, τότε προφανώς το μέγεθος του επιπέδου εισόδου θα είναι και αυτό ίσο με $d$. Στην συνέχεια θα μπορούσαμε να ορίσουμε ένα σύνολο $n$ απλών κρυφών επιπέδων τα οποία συνδέονται σειριακά μεταξύ τους, με αντίστοιχα μεγέθη $\lceil \frac{d}{i+1} \rceil$, όπου $i$ ο δείκτης που υποδεικνύει την θέση κάθε επιπέδου. Εάν το επίπεδο εισόδου κατέχει τον δίκτη $i = 0$, τότε το πρώτο κρυφό επίπεδο με δείκτη $i = 1$ θα έχει μέγεθος$\lceil \frac{d}{2} \rceil$, ενώ ακολουθώντας την ίδια λογική φτάνουμε μέχρι και το τελευταίο κρυφό επίπεδο με δείκτη $i = n$ το οποίο θα περιέχει $\lceil \frac{d}{n+1} \rceil$ νευρώνες.

- Επίπεδο Bottleneck: Το επίπεδο αυτό βρίσκεται αμέσως μετά τον Κωδικοποιητή, και συνδέεται με το τελευταίο του κρυφό επίπεδό[4]. Το μέγεθός του είναι ακόμη μικρότερο, για παράδειγμα θα μπορούσαμε να θέσουμε πλήθος νευρώνων ίσο με $\lceil \frac{d}{n+2} \rceil$, και στην ουσία μέσω αυτού ορίζεται το πλήθος των χαρακτηριστικών που προσπαθούμε να εξάγουμε από τα δεδομένα.

- Αποκωδικοποιητής: Το τελευταίο μέρος του Αυτοκωδικοποιητή περιέχει τα υπόλοιπα κρυφά επίπεδα του δικτύου καθώς και το επίπεδο εξόδου. Συνηθίζεται να ορίζουμε πλήθος κρυφών επιπέδων ίδιο με αυτό του Κωδικοποιητή, ενώ τα μεγέθη των επιμέρους επιπέδων συχνά αντικατοπτρίζουν τα μεγέθη των αντίστοιχων κρυφών επιπέδων του Κωδικοποιητή, με την διαφορά ότι παρατηρείται σταδιακή αύξηση αντί μείωσή τους. Για παράδειγμα, εάν αποδώσουμε και πάλι τους δείκτες $i \in \{1, ..., n\}$ στα κρυφά επίπεδα του Αποκωδικοποιητή, με τον δείκτη $i = 1$ να αντιστοιχεί στο πρώτο επίπεδό του, φτάνοντας έως και το τελευταίο κρυφό επίπεδο με δείκτη $i = n$, τότε θα μπορούσαμε να ορίσουμε το πλήθος νευρώνων του κάθε επιπέδου ως $\lceil \frac{d}{n+2-i} \rceil$. Τέλος, το $n$-οστό κρυφό επίπεδο του Αποκωδικοποιητή θα συνδέεται με το επίπεδο εξόδου, το μέγεθος του οποίου θα πρέπει υποχρεωτικά να ισούται με το μέγεθος του επιπέδου εισόδου, δηλαδή με την τιμή $d$.

Βάσει της παραπάνω αρχιτεκτονικής παρατηρούμε ότι οι Αυτοκωδικοποιητές αποτελούν στην ουσία ένα ιδιαίτερο είδος δικτύου, τα επίπεδα του οποίου είναι συμμετρικά ως προς το επίπεδο Bottleneck. Τα δεδομένα εισέρχονται στο δίκτυο μέσω του επιπέδου εισόδου, και στην συνέχεια η διαστασιμότητά τους συρρικνώνεται έως ότου να φτάσουν μέχρι και το Επίπεδο Bottleneck. Από εκεί και ύστερα το πλήθος των χαρακτηριστικών τους ξεκινάει ολοένα και να αυξάνεται ώσπου να φτάσουν στο επίπεδο εξόδου του δικτύου όπου θα έχουν πλέον αποκτήσει και πάλι την αρχική τους διαστασημότητα. Ποιά είναι όμως η λειτουργία κάθε μέρους του δικτύου, και πως αυτό μαθαίνει να «συμπιέζει» επιτυχώς τα δεδομένα;

[4]Πολλές φορές ωστόσο θεωρούμε ότι το επίπεδο αυτό αποτελεί επίσης μέρος του Κωδικοποιητή ως το τελευταίο του επίπεδο.

Όπως προαναφέρθηκε οι Αυτοκωδικοποιητές ανήκουν στην κατηγορία μοντέλων Μη-Επιβλεπόμενης Μάθησης. Κατά την εκπαίδευση ενός μοντέλου Αποκωδικοποιητή σκοπό αποτελεί η ανακατασκευή των δεδομένων εισόδου στην έξοδο του δικτύου. Θα ϑέλαμε δηλαδή ιδανικά, δοθέντος ενός διανύσματος εισόδου $\boldsymbol{x} \in \mathbb{R}^d$, η έξοδος $\tilde{\boldsymbol{x}} \in \mathbb{R}^d$ του δικτύου να ταυτίζεται με αυτό. Η «ταύτιση» αυτή των δύο διανυσμάτων εισόδου και εξόδου, είναι εύκολο να ποσοτικοποιηϑεί μέσω διαφόρων ειδών συναρτήσεων. Για παράδειγμα ϑα μπορούσαμε να μετρήσουμε την ευκλείδεια απόσταση μεταξύ των δύο διανυσμάτων, ορίζοντας έτσι το σφάλμα του δικτύου για ένα διάνυσμα εισόδου $\boldsymbol{x}$ ως $e = \| \boldsymbol{x} - \tilde{\boldsymbol{x}} \|_2$. Η τιμή $e$ ονομάζεται σφάλμα ανακατασκευής του στοιχείου $\boldsymbol{x}$. Ακολουϑώντας την ίδια λογική, δεδομένου ενός συνόλου δεδομένων αποτελούμενο από $n$ διανύσματα εισόδου $\boldsymbol{x}_i \in \mathbb{R}^d$, μπορούμε να ορίσουμε την συνάρτηση σφάλματος $J$ του δικτύου ως:

$$J(\boldsymbol{e}) = \frac{1}{2n} \sum_{i=1}^{n} e_i^2 = \frac{1}{2n} \sum_{i=1}^{n} \| \boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i \|_2^2 = \frac{1}{2n} \sum_{i=1}^{n} \sum_{j=1}^{d} (x_{ij} - \tilde{x}_{ij})^2$$

Εφόσον έχουμε ορίσει την συνάρτηση σφάλματος μπορούμε στην συνέχεια να χρησιμοποιήσουμε την κλασική μέϑοδο Backpropagation σε συνδυασμό με έναν αλγόριϑμο ελαχιστοποίησης, όπως ο αλγόριϑμος Κατάβασης Κλίσης, για να εκπαιδεύσουμε το δίκτυο. Εκτελώντας κάμποσες επαναλήψεις, το δίκτυο ϑα πρέπει να μάϑει να ανακατασκευάζει στην έξοδο τα διανύσματα τα οποία δέχεται στην είσοδο. Προκειμένου όμως τα δεδομένα να φτάσουν στην έξοδο του δικτύου, ϑα πρέπει πρώτα να περάσουν από όλα τα κρυφά επίπεδα του δικτύου, συμπεριλαμβανομένου και του επιπέδου Bottleneck, κατά το πέρασμα στο οποίο η διαστασημότητα των δεδομένων ϑα έχει πλέον συρρικνωϑεί κατά έναν μεγάλο βαϑμό. Συνεπώς το επίπεδο αυτό είναι κατά κάποιον τρόπο υποχρεωμένο να «μάϑει» να διατηρεί όσο το δυνατόν σημαντικότερη πληροφορία από τα δεδομένα, έτσι ώστε τα ανακατασκευασμένα διανύσματα εξόδου να παράγουν όλο και μικρότερο σφάλμα, δηλαδή να ταυτίζονται όλο και περισσότερο με την εκάστοτε είσοδο.

Θα λέγαμε λοιπόν ότι όπως υποδηλώνει και το όνομά του, η λειτουργία του Κωδικοποιητή του δικτύου συνίσταται στην «κωδικοποίηση» των δεδομένων ως μία αναπαράσταση χαμηλότερης διαστασιμότητας, η οποία ορίζεται μέσω του επιπέδου Bottleneck. Στην συνέχεια, ο Αποκωδικοποιητής δέχεται την αναπαράσταση αυτή, και επιχειρεί να «αποκωδικοποιήσει» τα δεδομένα, δηλαδή να τα επαναφέρει στην αρχική τους μορφή. Στο πρόβλημα της επιλογής χαρακτηριστικών που εξετάζουμε προφανώς δεν μας ενδιαφέρει η ανακατασκευή των δεδομένων, αλλά μονάχα το μέρος της κωδικοποιησής τους. Για αυτόν το λόγο το τμήμα του Αποκωδικοποιητή αφαιρείται από το δίκτυο μετά το πέρας της εκπαίδευσης, διατηρώντας μονάχα τα πρώτα δύο μέρη. Αυτό σημαίνει ότι το επίπεδο Bottleneck ϑα αποτελεί το νέο επίπεδο εξόδου του δικτύου, το οποίο πλέον ϑα εξάγει μία νέα αναπαράσταση των δεδομένων, ικανή να ενσωματώσει την ίδια αρχική πληροφορία σε ένα νέο χαμηλότερο πλήϑος διαστάσεων. Στην συνέχεια το «κωδικοποιημένο» αυτό σύνολο δεδομένων ϑα μπορούσε να χρησιμοποιηϑεί σε συνδυασμό με άλλα μοντέλα Μηχανικής Μάϑησης για την επίλυση περαιτέρω προβλημάτων.

Αφότου η προοπτική των Αυτοκωδικοποιητών αναγνωρίστηκε από την ευρύτερη επιστημονική κοινότητα, σύντομα άρχισαν να κάνουν την εμφάνισή τους διάφορες παραλλαγές και τροποποιήσεις των μοντέλων αυτών. Από τους «Denoising Autoencoders» [6] οι οποίοι επιχειρούν να ανακατασκευάσουν δεδομένα που έχουν υποστεί θόρυβο, μέχρι και τους «Variatonal Autoencoders» [7], οι οποίοι είναι σε θέση να αφομοιώσουν την πιθανοτική κατανομή των δεδομένων ενός συνόλου παράγοντας έτσι εντελώς νέα στοιχεία, οι Αυτοκωδικοποιητές ήταν πλέον ικανοί να αντιμετωπίσουν ένα ακόμη μεγαλύτερο εύρος προβλημάτων. Ένα από αυτά τα προβλήματα αποτελεί και η Ανίχνευση Ανωμαλιών (Anomaly Detection), για την οποία το μοντέλο θα πρέπει να μπορεί να «υποπτευθεί» ένα στοιχείο, όταν αυτό διαφέρει σε μεγάλο βαθμό από τα στοιχεία τα οποία έχει συναντήσει κατά την εκπαίδευσή του.

Για να εφαρμόσουμε έναν Αυτοκωδικοποιητή ως προς την επίλυση ενός τέτοιου προβλήματος δεν χρειάζεται να εκτελέσουμε καμία άλλη τροποποίηση των μοντέλων, παρά μόνο να παραβλέψουμε το βήμα αφαίρεσης του Αποκωδικοποιητή μετά την εκπαίδευση, καθώς η ανίχνευση ανωμαλιών χρήζει και των τριών μερών του δικτύου. Η διαδικασία έχει ως εξής: Εφόσον έχουμε εκπαιδεύσει το μοντέλο, διατηρώντας αυτή την φορά τον Αποκωδικοποιητή, πλέον στην έξοδο του δικτύου λαμβάνουμε την ανακατασκευή $\tilde{\boldsymbol{x}}$ της εισόδου $\boldsymbol{x}$. Όπως έχει προαναφερθεί, στα πλαίσια της εκπαίδευσης του μοντέλου ορίζουμε μία συνάρτηση υπολογισμού του σφάλματος ανακατασκευής $e$ το οποίο ποσοτικοποιεί την «διαφορά» μεταξύ των δύο διανυσμάτων $\boldsymbol{x}$ και $\tilde{\boldsymbol{x}}$. Είναι λογικό να υποθέσουμε ότι οι τιμές των σφαλμάτων που υπολογίζονται βάσει των δεδομένων εκπαίδευσης, καθώς και περαιτέρω δεδομένων τα οποία όμως προέρχονται από την ίδια κατανομή, θα είναι αρκετά χαμηλές εφόσον το δίκτυο έχει μάθει να ανακατασκευάζει επιτυχώς τα δεδομένα του συνόλου εκπαίδευσης. Από την άλλη, τροφοδοτώντας το δίκτυο με ένα άγνωστο στοιχείο εισόδου, το οποίο παράλληλα διαφέρει σε μεγάλο βαθμό από τα δεδομένα εκπαίδευσης, αναμένουμε ότι το σφάλμα ανακατασκευής θα είναι αρκετά υψηλότερο καθώς το δίκτυο δεν έχει εκπαιδευτεί ως προς την ανακατασκευή του.

Συνεπώς γίνεται αντιληπτό πώς μπορούμε να χρησιμοποιήσουμε το ίδιο το σφάλμα ανακατασκευής ως μία ένδειξη σχετικά με το εάν το στοιχείο στην είσοδο είναι «γνωστό» στο δίκτυο ή όχι. Εκπαιδεύοντας το μοντέλο βάσει ενός συνόλου δεδομένων τα οποία θεωρούνται αντιπροσωπευτικά της κατανομής που μας ενδιαφέρει, μπορούμε στην συνέχεια να χρησιμοποιήσουμε το σφάλμα κατασκευής για την ανίχνευση στοιχείων τα οποία δεν προέρχονται από την ίδια κατανομή, δηλαδή αποτελούν ανώμαλίες. Λόγω αυτής της ικανότητας των Αυτοκωδικοποιητών να προσαρμόζονται ως προς την αντιμετώπιση διαφόρων ειδών προβλημάτων, δικαίως κατέχουν σήμερα την θέση ενός από τα γνωστότερα είδη μοντέλων νευρωνικών δικτύων.

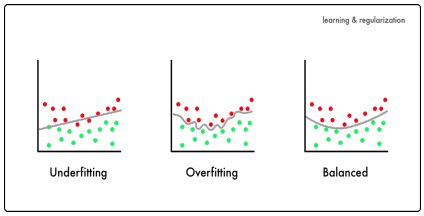
### 2.1.8 Τεχνικές Ομαλοποίησης

Τέλος θα αναφερθούμε σε ένα σύνολο τεχνικών που μπορούμε να χρησιμοποιήσουμε για να αυξήσουμε την ικανότητα γενίκευσης του δικτύου, δηλαδή την αποφυγή της υπερεκπαίδευσής του. Με τον όρο «Υπερεκπαίδευση» (Overfitting)

περιγράφουμε το φαινόμενο κατά το οποίο ένα μοντέλο Μηχανικής Μάθησης έχει μάθει να προσαρμόζεται τόσο καλά στα δεδομένα εκπαίδευσης, με αποτέλεσμα την μείωση της προβλεπτικής του ικανότητας πάνω σε άλλα σύνολα δεδομένων. Προφανώς θα θέλαμε ένα δίκτυο να μάθει να ταξινομεί με επιτυχία τα στοιχεία εκπαίδευσης, ιδιαίτερα όταν το σύνολο εκπαίδευσης είναι αρκετά μεγάλο και δεν περιέχει ιδιομορφίες. Καθώς όμως στην πραγματικότητα ένα τέτοιο σύνολο εκπαίδευσης είναι δύσκολο να κατασκευαστεί, τείνουμε να χρησιμοποιούμε ένα πλήθος μεθόδων οι οποίες επιτρέπουν στο δίκτυο να διατηρήσει την ικανότητα γενίκευσής του.

Από την άλλη, πολλές φορές προκύπτει το φαινόμενο της «Υποεκπαίδευσης» (Underfitting), δηλαδή το δίκτυο αδυνατεί να εκπαιδευτεί επιτυχώς, κατορθώνοντας χαμηλές επιδόσεις ακόμη και πάνω στο σύνολο εκπαίδευσης. Το πρόβλημα αυτό παρόλα αυτά αντιμετωπίζεται ευκολότερα, εκτελώντας περισσότερες επαναλήψεις κατά την εκπαίδευση και ρυθμίζοντας κατάλληλα τις τιμές των υπερπαραμέτρων του μοντέλου. Σε περίπτωση που το φαινόμενο της υποεκπαίδευσης επιμένει, τότε αυτό σημαίνει ότι είτε το σύνολο δεδομένων χρήζει περαιτέρω επεξεργασίας, είτε χρειαζόμαστε ένα διαφορετικό ισχυρότερο μοντέλο, ικανό να εμπεδώσει την πραγματική κατανομή των δεδομένων. Παρακάτω επικεντρωνόμαστε κυρίως στο φαινόμενο της Υπερεκπαίδευσης, παρουσιάζοντας μερικές τεχνικές εξάλειψής του.

Σχήμα 2.8: Τα φαινόμενα της Υπερεκπαίδευσης και Υποεκπαίδευσης




Πηγή: `https://towardsdatascience.com/8-simple-techniques-to-prevent-overfitting-4d443da2ef7d`

### Κανονικοποίηση των Δεδομένων

Η μέθοδος αυτή αφορά τα αριθμητικά και μόνο χαρακτηριστικά των δεδομένων. Σε περίπτωση που ορισμένα χαρακτηριστικά είναι κατηγορικά, ακόμη και εάν αυτά αναπαριστάνονται μέσω ακέραιων αριθμών, θα πρέπει να αποφύγουμε την κανονικοποίησή τους. Η κανονικοποίηση πρέπει να γίνεται ανα χαρακτηριστικό ξεχωριστά, λαμβάνοντας υπόψη όλα τα δεδομένα του συνόλου εκπαίδευσης. Δύο από τις πιο γνωστές μεθόδους κανονικοποίησης είναι η «Τυποποίηση» (Standardization) και η «0-1 Κανονικοποίηση» (0-1 Normalization). Στην πρώτη περίπτωση «τυποποιούμε» τα χαρακτηριστικά των δεδομένων, αφαιρόντας την μέση τιμή τους και διαιρώντας με την τυπική τους απόκλιση ως εξής:

$$x' = \frac{x - \mu}{\sigma}$$

Η διαδικασία αυτή μετασχηματίζει τις τιμές των δεδομένων, έτσι ώστε οι τιμές κάθε χαρακτηριστικού του συνόλου να ακολουθούν μία νέα κατανομή με μηδενική μέση τιμή και μοναδιαία τυπική απόκλιση. Από την άλλη η μέθοδος της «0-1 Κανονικοποίησης» προϋποθέτει την γνώση του εύρους τιμών των χαρακτηριστικών,

καθώς κάθε τιμή κανονικοποιείται αφαιρώντας την ελάχιστη τιμή και διαιρώντας με την διαφορά μεταξύ μέγιστης και ελάχιστης τιμής:

$$x' = \frac{x - m}{M - m}$$

όπου $m$ και $M$ η ελάχιστη και η μέγιστη τιμή αντίστοιχα του εύρους τιμών του $x$. Σε περίπτωση που δεν υπάρχει εκ των προτέρων γνώση του εύρους τιμών μίας μεταβλητής, τότε μπορούμε να χρησιμοποιήσουμε την ελάχιστη/μέγιστη τιμή που συναντάμε εντός του συνόλου εκπαίδευσης. Έχει αποδειχθεί πειραματικά ότι σε πολλές περιπτώσεις οι δύο αυτές μέθοδοι κανονικοποίησης επιτρέπουν στο μοντέλο να κατορθώσει υψηλότερες επιδόσεις.

### $L_1/L_2$ Ομαλοποίηση

Μία από τις παλαιότερες τεχνικές ομαλοποίησης αφορά την χρήση ενός όρου $\Omega$, ο οποίος πολλαπλασιασμένος με μία παράμετρο $\lambda \in [0, 1]$ προστίθεται στην συνάρτηση σφάλματος προς ελαχιστοποίηση. Η τιμή της παραμέτρου $\lambda$ καθορίζεται από εμάς με σκοπό τον έλεγχο του «βάρους» που έχει ο όρος $\Omega$ πάνω στην συνάρτηση σφάλματος $J$. Επομένως η νέα συνάρτηση σφάλματος $J'$ λαμβάνει την εξής μορφή:

$$J' = J + \lambda \cdot \Omega$$

Στα πλαίσια της εκπαίδευσης των νευρωνικών δικτύων ο όρος ομαλοποίησης $\Omega$ αποσκοπεί στον περιορισμό των τιμών των συναπτικών βαρών τους, καθώς υψηλές τιμές ενδέχεται να οδηγήσουν σε ασταθή μοντέλα και στο φαινόμενο της Υπερεκπαίδευσης. Στην ουσία ο όρος $\Omega$ δεν είναι παρά μονάχα μία συνάρτηση η οποία δέχεται τις τιμές των συναπτικών βαρών ενός επιπέδου του δικτύου, και βάσει αυτών εξάγει μία τιμή. Δεδομένου ενός επιπέδου $i$ του δικτύου αποτελούμενο από $n$ νευρώνες, στον καθένα από τους οποίους αντιστοιχουν και από $m$ συναπτικά βάρη, μπορούμε να οργανώσουμε τις τιμές των συναπτικών βαρών του επιπέδου σε έναν πίνακα $\boldsymbol{W}_i$ διαστάσεων $n \times m$. Εάν τώρα συμβολίσουμε ως $I$ το σύνολο το οποίο περιέχει τους δείκτες των επιπέδων του δικτύου πάνω στα οποία θα θέλαμε να εφαρμόσουμε ομαλοποίηση, η συνάρτηση σφάλματος $J'$ λαμβάνει την εξής τελική της μορφή:

$$J' = J + \lambda \cdot \sum_{i \in I} \Omega(\boldsymbol{W}_i)$$

Συνεπώς, η συνάρτηση $\Omega$ θα πρέπει να δέχεται έναν οποιονδήποτε δισδιάστατο πίνακα $\boldsymbol{W}_i$, και βάσει αυτού να υπολογίζει μία και μόνο τιμή, η οποία θα είναι ικανή να εκφράσει το μέγεθος των συναπτικών βαρών του. Αυτό μπορεί να επιτευχθεί μέσω της διανυσματικής νόρμας $\| \cdot \|_p$. Συγκεκριμένα ορίζουμε:

$$\Omega(\boldsymbol{W}_i) = \sum_{j=1}^{n} \| \boldsymbol{w}_{ij} \|_p^p \ , \text{ όπου } \boldsymbol{w}_{ij} \in \mathbb{R}^m$$

Η παράμετρος $p \in \{1, 2\}$ καθορίζει τον τύπο της νόρμας που χρησιμοποιούμε. Αναλόγως της τιμής $p$ κάνουμε λόγο για $L_1$ ή $L_2$ Ομαλοποίηση [8]. Η $L_1$ ομαλοποίηση, η αλλιώς γνωστή με το όνομα «Παλινδρόμηση Λάσσο» (Lasso Regression),

χρησιμοποιεί την $L_1$ νόρμα με σκοπό τον περιορισμό των τιμών των συναπτικών βαρών, με αποτέλεσμα η συνάρτηση $\Omega$ να λάβει την εξής μορφή:

$$\Omega(\boldsymbol{W}_i) = \sum_{j=1}^{n} \parallel \boldsymbol{w}_{ij} \parallel_1 = \sum_{j=1}^{n}\sum_{k=1}^{m} \mid w_{ijk} \mid$$

Λόγω του ότι κάθε τιμή συναπτικού βάρους «τιμωρείται» ισάξια ανεξαρτήτως του μεγέθους της, η χρήση της νόρμας $L_1$ οδηγεί συνήθως σε αραιές λύσεις, καθώς συναπτικά βάρη τα οποία κατέχουν ήδη μικρές τιμές τείνουν να μηδενίζονται. Από την άλλη θέτοντας $p = 2$ εφαρμόζουμε $L_2$ Ομαλοποίηση, ή αλλιώς «Ομαλοποίηση Ρίντζ» (Ridge Regression), με την συνάρτηση $\Omega$ να λαμβάνει την εξής μορφή:

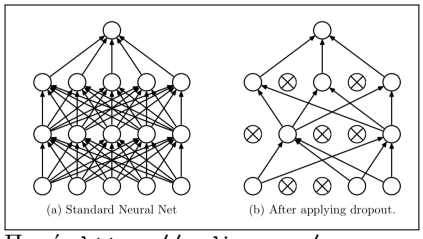
$$\Omega(\boldsymbol{W}_i) = \sum_{j=1}^{n} \parallel \boldsymbol{w}_{ij} \parallel_2^2 = \sum_{j=1}^{n}\sum_{k=1}^{m} w_{ijk}^2$$

Θα λέγαμε γενικά ότι η χρήση της νόρμας $L_2$ θεωρείται περισσότερο ισορροπημένη, καθώς μέσω των τετραγώνων των τιμών δίνεται περισσότερη βαρύτητα στα συναπτικά βάρη τα οποία κατέχουν υψηλές τιμές, ενώ οι τροποποιήσεις που υφίστανται τα συναπτικά βάρη χαμηλότερων τιμών είναι περισσότερο ήπιες σε σχέση με την περίπτωση χρήσης της $L_1$ νόρμας. Ωστόσο και οι δύο τεχνικές είναι ικανές να συμβάλουν στην αντιμετώπιση του προβλήματος της Υπερεκπαίδευσης, καθώς και σε διάφορα άλλα είδη προβλημάτων όπως για παράδειγμα το φαινόμενο της «Εκρηγνυόμενης Κλίσης» (Exploding Gradient), για την εξάλειψη του οποίου οι χαμηλές τιμές συναπτικών βαρών κατέχουν έναν σημαντικό ρόλο.

### Η Μέθοδος Dropout

Η μέθοδος Dropout [9] αποτελεί πλεόν μία από τις πιο διαδεδομένες τεχνικές ομαλοποίησης και εφαρμόζεται κυρίως στα κρυφά επίπεδα του δικτύου μέσω μία παραμέτρου πιθανότητας $p \in [0, 1]$, βάσει της οποίας αφαιρείται ένα τυχαίο ποσοστό των συνάψεων που ενώνουν τους νευρώνες του εκάστοτε κρυφού επιπέδου με τους νευρώνες του αμέσως προηγούμενου επιπέδου. Για παράδειγμα εφαρμόζοντας την μέθοδο Dropout με πιθανότητα $p = 0.2$ σε ένα κρυφό επίπεδο του δικτύου, στατιστικά αναμένουμε ότι το επίπεδο θα διατηρησει μονάχα το 80% των συνάψεών του. Θα πρέπει να τονίσουμε ότι αυτή η αφαίρεση των συνάψεων είναι προσωρινή, καθώς η μέθοδος θα πρέπει να εφαρμόζεται μονάχα κατά την διάρκεια της εκπαίδευσης του δικτύου. Όσο αφορά την παράμετρο $p$ συνηθίζουμε να ορίζουμε την τιμή της στο διάστημα $[0.1, 0.5]$. Πολλάκις στο παρελθόν εχει αποδειχθεί πειραματικά ότι η τεχνική αυτή μπορεί να αυξήσει σε μεγάλο βαθμό την ικανότητα γενίκευσης του δικτύου.

Σχήμα 2.9: Εφαρμογή της μεθόδου Dropout.




Πηγή: `https://medium.com/konvergen/understanding-dropout-ddb60c9f98aa`

## 2.2 Τα Συστήματα Hierarchical Temporal Memory

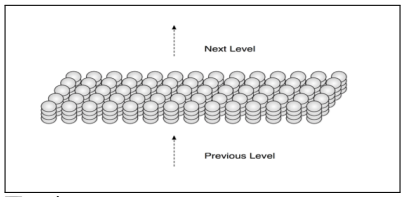
Τα συστήματα Hierarchical Temporal Memory (HTM) περιγράφτηκαν για πρώτη φορά το 2004 από τον Τζέφ Χόκινς στο βιβλίο του με τίτλο «On Intelligence» [10]. Πρόκειται για ένα καινοτόμο μοντέλο Μηχανικής Μάθησης του οποίου η λειτουργία βασίζεται εξ ολοκλήρου πάνω σε ένα μέρος του εγκεφαλικού φλοιού, τον «νεοφλοιό» (neocortex), ο οποίος είναι υπεύθυνος για ένα σύνολο βασικών εγκεφαλικών δραστηριοτήτων, όπως για παράδειγμα είναι η αντίληψη των αισθήσεων, η κατανόηση της γλώσσας και ο έλεγχος των κινήσεων [11]. Λόγω αυτής της έντονης επιρροής από την βιολογία, τα συστήματα HTM θα δούμε ότι διαφέρουν κατά έναν πολύ μεγάλο βαθμό από τα περισσότερα μοντέλα Μηχανικής Μάθησης.

### 2.2.1 Η Αρχιτεκτονική των Συστημάτων HTM

Κάθε σύστημα HTM αποτελείται από ένα πλήθος «περιοχών» (regions) ή αλλιώς «επιπέδων» (levels)[5] τα οποία οργανώνονται βάσει μίας ιεραρχίας. Η οργάνωση των περιοχών θυμίζει αρκετά την οργάνωση των επιπέδων των κλασικών νευρωνικών δικτύων: Η πρώτη περιοχή δέχεται ένα πρότυπο εισόδου βάσει του οποίου παράγει μία έξοδο, η οποία έπειτα τροφοδοτείται στην αμέσως επόμενη περιοχή που ορίζει η ιεραρχία. Η διαδικασία αυτή επαναλαμβάνεται έως ότου η τελευταία περιοχή να παράγει την τελική έξοδο του δικτύου. Αναλόγως της θέσης της στην ιεραρχία κάθε περιοχή «μαθαίνει» να αναγνωρίζει και διαφορετικά χαρακτηριστικά της εισόδου. Περιοχές οι οποίες βρίσκονται χαμηλά στην ιεραρχία σχετίζονται με την εκμάθηση βασικών και γενικότερων χαρακτηριστικών, ενώ όσο πλησιάζουμε τις ανώτερες περιοχές τα χαρακτηριστικά γίνονται ολοένα και πιο αφηρημένα.

Για να κατανοήσουμε την λειτουργία των περιοχών θα πρέπει να εισάγουμε δύο ακόμη έννοιες, αυτές της «στήλης» (column) και του «κυττάρου» (cell). Μία περιοχή ουσιαστικά αποτελείται από ένα πλήθος στηλών, οι οποίες συνηθίζεται (όχι απαραίτητα) να οργανώνονται σε έναν δισδιάστατο πίνακα. Κάθε στήλη με την σειρά της αποτελείται από ένα πλήθος κυττάρων τα οποία ενδέχεται να είναι συνδεδεμένα με άλλα κύτταρα εντός της ίδιας περιοχής. Οι συνδέσεις αυτές πραγματοποιούνται βάσει ορισμένων «συνάψεων» (synapses), οι οποίες ανήκουν στα «τμημάτα δενδρίτη» (dendrite segments) των κυττάρων. Γενικά υπάρχουν δύο είδη

Σχήμα 2.10: Μία περιοχή ενός δικτύου HTM αποτελούμενη από 70 στήλες τεσσάρων κυττάρων




Πηγή: `Hierarchical Temporal Memory Whitepaper by Numenta`

[5] Αν και ένα επίπεδο μπορεί θεωρητικά να εμπεριέχει περισσότερες της μίας περιοχής, συνηθίζεται να χρησιμοποιούμε μονάχα μία περιοχή ανά επίπεδο με αποτέλεσμα οι δύο έννοιες να είναι πρακτικά ισοδύναμες.

τμημάτων δενδρίτη, αναλόγως του τύπου σύνδεσης που πραγματοποιούν:

1. Κεντρικό τμήμα δενδρίτη (Proximal dendrite segment): Σε αυτό το είδος τμήματος δενδρίτη ανήκουν οι συνάψεις οι οποίες συνδέουν τις στήλες μίας περιοχής με την είσοδο σε αυτή, είτε αυτή προέρχεται από την αμέσως προηγούμενη περιοχή στην ιεραρχία είτε απευθείας από μία πηγή, π.χ. από κάποιον σένσορα. Παρατηρούμε λοιπόν ότι οι στήλες αντιμετωπίζονται ως ενιαίες υπολογιστικές μονάδες, σε κάθε μία από τις οποίες αντιστοιχεί και από ένα ξεχωριστό κεντρικό τμήμα δενδρίτη, το οποίο με την σειρά του περιέχει ένα ξεχωριστό σύνολο συνάψεων. Αυτό σημαίνει ότι κάθε στήλη ενδέχεται να είναι συνδεδεμένη με ένα ελαφρώς διαφορετικό τμήμα της εισόδου στην περιοχή.

2. Περιφερικό τμήμα δενδρίτη (Distal dendrite segment): Κάθε κύτταρο διαθέτει περισσότερα του ενός περιφερικά τμήματα δενδρίτη, οι συνάψεις των οποίων το συνδέουν με άλλα κύτταρα εντός της ίδιας περιοχής. Συνεπώς γίνεται φανερό ότι το πλήθος των περιφερικών τμημάτων δενδρίτη είναι αρκετά μεγαλύτερο από το πλήθος των κεντρικών. Για παράδειγμα, εάν μία περιοχή διαθέτει 100 στήλες των 10 κυττάρων, στο καθένα από τα οποία αντιστοιχούν 5 περιφερικά τμήματα δενδρίτη, τότε το πλήθος των κεντρικών τμημάτων δενδρίτη είναι μόλις 100, ενώ το πλήθος των περιφερικών ισούται με $100 \cdot 10 \cdot 5 = 5,000$.

Αναφορικά με τις συνάψεις των τμημάτων δενδρίτη, σε κάθε μία από αυτές αντιστοιχεί και από ένα δυαδικό βάρος, ή πιο απλά κάθε σύναψη θα θεωρείται «ενεργή» (active) ή «ανενεργή» (inactive). Επιπλέον σε κάθε σύναψη αντιστοιχεί και μία ακόμη τιμή, γνωστή και ως «τιμή μονιμότητας» (permanence value), η οποία στην ουσία καθορίζει κατά πόσο μία σύναψη θεωρείται «συνδεδεμένη» (connected) ή όχι. Αργότερα θα εμβαθύνουμε περαιτέρω στην σημασία των παραπάνω ιδιοτήτων των συνάψεων. Για τώρα αρκεί μονάχα να θυμόμαστε ότι ένα σύστημα HTM ορίζεται ως μία ιεραρχία περιοχών, κάθε μία από τις οποίες απαρτίζεται από ένα πλήθος αλληλο-συνδεόμενων κυττάρων οργανωμένων σε στήλες, οι οποίες με την σειρά τους συνδέονται με την εκάστοτε είσοδο που τροφοδοτείται στην περιοχή.

### 2.2.2 Οι Αραιές Κατανεμημένες Αναπαραστάσεις

Σε αντίθεση με τα περισσότερα μοντέλα Μηχανικής Μάθησης, στα πλαίσια των οποίων η είσοδος αναπαριστάνεται ως ένα διάνυσμα χαρακτηριστικών $\boldsymbol{x} \in \mathbb{R}^d$, τα συστήματα HTM απαιτούν από την είσοδο να κατέχει την μορφή δυαδικής συμβολοσειράς, δηλαδή $\boldsymbol{x} \in \{0,1\}^n$, όπου $n$ ένα προκαθορισμένο μήκος. Επιπλέον, θα θέλαμε ιδανικά το δίκτυο να διαχειρίζεται συμβολοσειρές οι οποίες είναι «αραιές» (sparse) και «κατανεμημένες» (distributed). Στα πλαίσια των συστημάτων HTM, οι δυαδικές συμβολοσειρές οι οποίες ικανοποιούν τις δύο αυτές προϋποθέσεις αποκαλούνται «Αραιές Κατανεμημένες Αναπαραστάσεις» (Sparse Distributed Representations - SDRs). Παρακάτω αναφερόμαστε λεπτομερέστερα στις δύο αυτές ιδιότητες, της «αραιής» και «κατανεμημένης» συμβολοσειράς:

1. Αραιή: Με τον όρο αυτό, εννοούμε την συμβολοσειρά της οποίας το πλήθος των τιμών «1», ή αλλιώς των ενεργών bit, είναι κατά ένα μεγάλο βαθμό μικρότερο από το πλήθος των τιμων «0», δηλαδή των ανενεργών bit. Έχει αποδειχτεί ότι ένα σύστημα HTM είναι αρκετά πιο ανθεκτικό στον θόρυβο όταν αυτό διαχειρίζεται αραιές συμβολοσειρές [12, 13]. Παράλληλα δεν θα πρέπει να υπάρχει φόβος σχετικά με το συνολικό πλήθος όλων των διαφορετικών δυαδικών συμβολοσειρών που μπορούμε να κατασκευάσουμε χρησιμοποιώντας μία συγκεκριμένη τιμή μήκους $n$, καθώς ακόμη και για υπερβολικά αραιές συμβολοσειρές το μέγεθος αυτό παραμένει εξαιρετικά μεγάλο. Συγκεκριμένα για συμβολοσειρές μήκους $n$ με πλήθος ενεργών bit $w \leq n$, μπορούμε να κατασκευάσουμε συνολικά $C(n, w)$ διαφορετικές συμβολοσειρές:

   $$C(n, w) = \binom{n}{w} = \frac{n!}{w!(n-w)!}$$

   Για παράδειγμα χρησιμοποιώντας μήκος $n = 200$ και ποσοστό ενεργών bit μόλις 2% ($w = 4$), μπορούμε να κατασκευάσουμε $C(200, 4) = 64,684,950$ διαφορετικές συμβολοσειρές.

2. Κατανεμημένη: Όταν λέμε ότι οι συμβολοσειρές θα πρέπει να είναι κατανεμημένες τότε αυτό σημαίνει ότι για κάποιο σύνολο συμβολοσειρών μήκους $n$ θα πρέπει όλα τα διαφορετικά bit να είναι ενεργά με περίπου την ίδια συχνότητα. Θέλουμε δηλαδή να μην υπάρχει κάποια θέση $i \in \{1, ..., n\}$, η οποία θα δέχεται πάντοτε την τιμή «0», καθώς έτσι θα έχανε το νόημά της. Οι είσοδοι άρα θα πρέπει να κωδικοποιούνται κατά τέτοιο τρόπο, έτσι ώστε τα εκάστοτε προτύπα εισόδου να είναι ικανά να «καλύψουν» από κοινού ένα μεγάλο τμήμα του χώρου των δυαδικών συμβολοσειρών $\{0, 1\}^n$. Θα ήταν επομένως ορθότερο να πούμε ότι ο όρος αυτός περιγράφει ένα σύνολο συμβολοσειρών, και όχι κάθε μεμονωμένη συμβολοσειρά ξεχωριστά.

Πως όμως μπορούμε να αναπαραστήσουμε την είσοδο σε μία τέτοια μορφή; Γενικά, κάθε συνάρτηση SDR κωδικοποίησης θα πρέπει να διέπεται από τέσσερεις βασικές αρχές [14]:

1. Σημασιολογικά όμοια δεδομένα θα πρέπει να αντιστοιχούν σε παρόμοιες κωδικοποιήσεις, για παράδειγμα η τιμή της απόστασης Hamming μεταξύ τους να είναι χαμηλή.

2. Η ίδια είσοδος θα πρέπει πάντοτε να παράγει την ίδια έξοδο.

3. Η έξοδος θα πρέπει πάντοτε να κατέχει το ίδιο μήκος για οποιαδήποτε είσοδο.

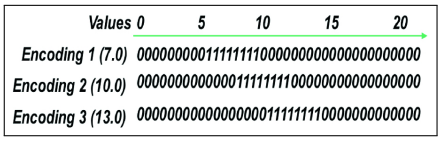
4. Η έξοδος θα πρέπει πάντοτε να διατηρεί σχετικά σταθερό και αραιό ποσοστό ενεργών bit.

Ως παράδειγμα θα αναφέρουμε μία απλή συνάρτηση κωδικοποίησης πραγματικών τιμών, την οποία όπως θα δούμε αργότερα πρόκειται να χρησιμοποιήσουμε και στην πράξη. Έστω ότι γνωρίζουμε εκ των προτέρων το σύνολο $[m, M]$ στο

οποίο ανήκουν οι τιμές ενός χαρακτηριστικού των διανυσμάτων εισόδου. Συνεπώς το διάστημα τιμών υπολογίζεται ως $range = M - m$. Στην συνέχεια ορίζουμε το μήκος $n$ των συμβολοσειρών που πρόκειται να κατασκευάσουμε, καθώς και το πλήθος των ενεργών bit $w \leq n$. Μέσω των δύο αυτών τιμών, ο κωδικοποιητής ορίζει ένα σύνολο «κάδων» (buckets) $B = \{b_1, ..., b_k\}$, όπου $k = n - w + 1$, σε κάθε έναν από τους οποίους αντιστοιχεί τις τιμές του διαστήματος $[m, M]$. Συγκεκριμένα σε κάθε κάδο $b_i \in B$ θα αντιστοιχούν οι τιμές του αντίστοιχου διαστήματος $[m + (i-1) \cdot \frac{range}{k}, m + i \cdot \frac{range}{k}]$. Για παράδειγμα εάν ισχύει ότι $(m, M) = (0, 100)$ και $k = 10$, τότε ο πρώτος κάδος θα περιέχει τις τιμές του διαστήματος $[0, 10]$, ο δεύτερος θα περιέχει τις τιμές στο διάστημα $(10, 20]$ και ούτω καθεξής.

Κάθε κάδος $b_i$ αναπαριστάνει μία διαφορετική συμβολοσειρά $x_i$. Στα πλαίσια της συνάρτησης κωδικοποίησης που περιγράφουμε κάθε συμβολοσειρά $x_i$ θα αποτελείται από ένα πλήθος $n$ bit, εντός των οποίων συναντάμε μία ενιαία ακολουθία $w$ ενεργών bit. Ο κάδος $b_1$ θα αντιστοιχεί για παράδειγμα στην συμβολοσειρά $x_1$, η ακολουθία ενεργών bit της οποίας θα ξεκινάει από την πρώτη θέσης, ενώ για κάθε επόμενο κάδο $b_{i>1}$ η ακολουθία αυτή θα μετατοπίζεται κατά μία θέση δεξιά εντός της αντίστοιχης συμβολοσειράς. Η διαδικασία γίνεται ευκολότερα κατανοητή μέσω του Σχήματος 2.11.

Σχήμα 2.11: Κωδικοποίηση τιμών στο διάστημα $[0, 20]$ με κωδικοποιητή παραμέτρων $(n, w) = (38, 8)$.




Πηγή: https://link.springer.com/chapter/10.1007/978-3-030-14524-8_13

Παρατηρούμε λοιπόν ότι τιμές οι οποίες βρίσκονται πολύ κοντά μεταξύ τους είναι πολύ πιθανό να αντιστοιχούν στον ίδιο κάδο, και άρα κατ΄ επέκταση να αναπαρίστανται μέσω της ίδιας συμβολοσειράς, ενώ τιμές οι οποίες βρίσκονται σχετικά κοντά μεταξύ τους ενδέχεται να αντιστοιχούν σε κοντινούς κάδους με αποτέλεσμα ναι μεν οι αναπαραστάσεις τους να διαφέρουν, αλλά παράλληλα να μοιράζονται αρκετά κοινά ενεργά bit. Από την άλλη όσες τιμές έχουν μεγάλη διαφορά μεταξύ τους θα αντιστοιχούν σε απομακρυσμένους κάδους, γεγονός το οποίο σημαίνει ότι θα μοιράζονται ελάχιστα ή ακόμη και κανένα ενεργό bit. Μέσω αυτής της διαδικασίας η συνάρτηση κωδικοποίησης επομένως αποτυπώνει στις συμβολοσειρές που κατασκευάζει τις σχέσεις μεταξύ των αρχικών τιμών του εκάστοτε χαρακτηριστικού.

Αφότου κάθε τιμή $x_i$ ενός στοιχείου $\boldsymbol{x} \in \mathbb{R}^d$ έχει κωδικοποιηθεί ως μία δυαδική συμβολοσειρά $s_i$, στην συνέχεια οι συμβολοσειρές αυτές συνενώνονται σε μία ενιαία συμβολοσειρά $s$ μήκους $n = \sum_{i=1}^{d} n_i$, όπου $n_i$ το μήκος της αντίστοιχης συμβολοσειράς $s_i$. Η συμβολοσειρά $s$ αποτελεί και την τελική αναπαράσταση του στοιχείου $\boldsymbol{x}$ βάσει της οποίας στην συνέχεια τροφοδοτείται στο σύστημα HTM. Η συνένωση των επιμέρους συμβολοσειρών $s_i$ μπορεί να πραγματοποιηθεί με την σειρά που εμείς επιθυμούμε, ενώ παράλληλα η τελική συμβολοσειρά $s$ μπορεί να οργανωθεί σε $n$-διάστατους πίνακες, αποδίδοντας κατά αυτόν τον τρόπο στην είσοδο μία επιπλέον τοπολογική πληροφορία, εάν κρίνουμε πως αυτή είναι απαραίτητη.

### 2.2.3 Ο Αλγόριθμος Spatial Pooler

Οι δύο βασικοί αλγόριθμοι που διέπουν την λειτουργία κάθε συστήματος HTM είναι οι «Spatial Pooler» και «Temporal Pooler». Κάθε περιοχή είναι στην ουσία υπεύθυνη για την εκτέλεση των δύο αυτών αλγορίθμων. Ξεκινώντας με την εκτέλεση του Spatial και έπειτα του Temporal Pooler –καθώς η έξοδος του πρώτου αποτελεί την είσοδο του δεύτερου– μία περιοχή παράγει την έξοδό της η οποία στην συνέχεια είτε τροφοδοτείται ως είσοδος στην αμέσως επόμενη περιοχή βάσει της ιεραρχίας, είτε αποτελεί την τελική έξοδο του συστήματος. Σε αυτό το μέρος θα αναλύσουμε τον πρώτο από τους δύο αλγορίθμους, ευθύνη του οποίου αποτελεί η αναγνώριση της όποιας τοπολογικής πληροφορίας εμπεριέχεται στην είσοδο, καθώς και η μετατροπή της σε μία SDR συμβολοσειρά. Αυτό σημαίνει ότι ακόμη και εάν η είσοδος δεν έχει κωδικοποιηθεί όπως πρέπει, ο αλγόριθμος αναλαμβάνει για αυτό. Ωστόσο η εκ των προτέρων κωδικοποίηση της εισόδου ως μία SDR συμβολοσειρά συνιστάται σε κάθε περίπτωση με σκοπό την ομαλότερη λειτουργία του συστήματος.

Θα μπορούσαμε να πούμε ότι ο αλγόριθμος Spatial Pooler εκτελεί στην ουσία τρία βήματα:

1. Υπολογισμός των τιμών «επικάλυψης» (overlap) μεταξύ εισόδου και στηλών.
2. Ενεργοποίηση των στηλών.
3. Φάση εκπαίδευσης του αλγορίθμου.

Τα πρώτα δύο βήματα του αλγορίθμου σχετίζονται με τον υπολογισμό της εξόδου, ενώ το τελευταίο βήμα έχει να κάνει με την εκπαίδευση του συστήματος, η οποία αφορά την ανανέωση των τιμών μονιμότητας των συνάψεων που ανήκουν στα κεντρικά τμήματα δενδρίτη των στηλών της περιοχής, καθώς και την ανανέωση των τιμών ορισμένων εσωτερικών μεταβλητών που διέπουν την λειτουργία του αλγορίθμου. Παρακάτω παρουσιάζονται τα τρία αυτά βήματα λεπτομερέστερα.

#### 1. Υπολογισμός των τιμών επικάλυψης

Σκοπός του αλγορίθμου Spatial Pooler είναι η μετατροπή της εισόδου της περιοχής σε μία SDR συμβολοσειρά. Αναφορικά με την κατανόηση αυτής της διαδικασίας είναι σημαντικό να τονίσουμε ότι στα πλαίσια της εκτέλεσης του αλγορίθμου Spatial Pooler ενδιαφερόμαστε αποκλειστικά και μόνο για τις συνάψεις των κεντρικών τμημάτων δενδρίτη, μέσω των οποίων κάθε στήλη της περιοχής συνδέεται με ένα υποσύνολο των bit της συμβολοσειράς εισόδου. Αυτό οφείλεται στο γεγονός ότι ο αλγόριθμος αυτός διαχειρίζεται τις στήλες ως ξεχωριστές υπολογιστικές μονάδες, αγνοώντας την λειτουργία των επιμέρους κυττάρων που τις απαρτίζουν, η οποία όπως θα δούμε αργότερα λαμβάνεται υπόψη από τον αλγόριθμο Temporal Pooler.

Όσο αφορά την τιμή επικάλυψης αυτή υπολογίζεται βάσει δύο συμβολοσειρών ίδιου μήκους, και συγκεκριμένα ορίζεται ως το πλήθος των ενεργών bit τα οποία είναι κοινά και για τις δύο συμβολοσειρές. Για παράδειγμα, η επικάλυψη

των συμβολοσειρών «0011» και «1011» είναι ίση με την τιμή δύο, λόγω των δύο κοινών ενεργών bit στην τρίτη και τέταρτη θέση των συμβολοσειρών. Αυτό το βήμα του αλγορίθμου αποσκοπεί μονάχα στον υπολογισμό των τιμών επικάλυψης μεταξύ της συμβολοσειράς της εισόδου –ή ορθότερα ενός μέρους αυτής– και των συμβολοσειρών που ορίζονται μέσω των συνάψεων κάθε στήλης.

Έστω ότι η είσοδος σε μία περιοχή αποτελεί μία συμβολοσειρά μήκους $n$. Όπως προαναφέρθηκε, σε κάθε στήλη $i$ της περιοχής αντιστοιχεί και από ένα κεντρικό τμήμα δενδρίτη το οποίο περιέχει ένα πλήθος συνάψεων $n_i \leq n$ μεταξύ της στήλης και των bit της εισόδου. Συνηθίζεται να ορίζουμε $n_i < n$, έτσι ώστε κάθε στήλη $i$ να συνδέεται με ένα ξεχωριστό υποσύνολο των bit της συμβολοσειράς εισόδου. Σε κάθε σύναψη αντιστοιχεί και από μία τιμή μονιμότητας $p \in [0, 1]$ η οποία μαζί με μία τιμή κατωφλιού[6] $\theta_p$ καθορίζουν εάν η σύναψη θεωρείται συνδεδεμένη ή όχι. Επιπλέον, εάν μία σύναψη συνδέεται με ένα ενεργό bit εισόδου, τότε η σύναψη αυτή θεωρείται ενεργή, ενώ μία συνδεδεμένη και ενεργή σύναψη θεωρείται ενεργοποιημένη:

- Η σύναψη $j$ μίας στήλης $i$ θα λέμε ότι είναι συνδεδεμένη εάν ισχύει ότι $p_{ij} \geq \theta_p$. Σε διαφορετική περίπτωση, η σύναψη θεωρείται μη-συνδεδεμένη.
- Μία σύναψη θα λέμε ότι είναι ενεργή εάν αυτή συνδέεται με ένα ενεργό bit εισόδου. Σε διαφορετική περίπτωση, η σύναψη θεωρείται ανενεργή.
- Μία σύναψη θα λέμε ότι είναι ενεργοποιημένη αν και μόνο αν η σύναψη είναι ταυτόχρονα ενεργή και συνδεδεμένη.

Η συμβολοσειρά μέσω της οποίας αναπαριστάνεται μία στήλη κατασκευάζεται βάσει των συνάψεων του κεντρικού τμήμα δενδρίτη που της αντιστοιχεί. Με τιμή «1» συμβολίζουμε τις συνδεδεμένες συνάψεις ενώ η τιμή «0» χρησιμοποιείται για τις υπόλοιπες. Εάν επομένως σε μία στήλη $i$ αντιστοιχεί ένα κεντρικό τμήμα δενδρίτη $n_i$ συνάψεων, τότε μπορούμε να την αναπαραστήσουμε ως μία συμβολοσειρά μήκους $n_i$. Εφόσον κάθε σύναψη αντιστοιχεί και σε ένα διαφορετικό bit εισόδου, μπορούμε πλέον να υπολογίσουμε την επικάλυψη της. Παρακάτω δίνουμε ένα απλό παράδειγμα υπολογισμού της τιμής επικάλυψης μίας στήλης.

Έστω μία συμβολοσειρά εισόδου $x = 1100100001$, και μία στήλη $i$ πέντε συνάψεων με αντίστοιχες τιμές μονιμότητας $\boldsymbol{p}_i = [0.8, 0.1, 0.72, 0.23, 0.9]$. Έστω ότι κάθε μία από αυτές τις συνάψεις αντιστοιχεί και σε ένα από τα πρώτα πέντε bit της εισόδου, δηλαδή στην συμβολοσειρά $x_i = 11001$. Δεδομένου προκαθορισμένου κατωφλιού $\theta_p = 0.5$, η συμβολοσειρά η οποία αναπαριστάνει την στήλη $i$ υπολογίζεται ως $s_i = 10101$. Βάσει των συμβολοσειρών $s_i$ και $x_i$ μπορούμε πλέον εύκολα να υπολογίσουμε την επικάλυψη $o_i$ της στήλης ίση με την τιμή δύο, καθώς το πρώτο και το τελευταίο αποτελούν τα μόνα δύο κοινά ενεργά bit ανάμεσα στις δύο συμβολοσειρές. Με απλά λόγια, ο γενικός κανόνας υπολογισμού της επικάλυψης είναι ο εξής: Η τιμή επικάλυψης μίας στήλης $i$ ως προς μία συμβολοσειρά εισόδου $x$ υπολογίζεται ως το πλήθος των ενεργοποιημένων συνάψεων της στήλης $i$ ως προς την συμβολοσειρά $x$.

[6] Βλέπε *Η Βιβλιοθήκη NuPIC / Η κλάση SpatialPooler / synPermConnected*.

Η παραπάνω διαδικασία επαναλαμβάνεται για κάθε στήλη $i$ της περιοχής ούτως ώστε να υπολογίσουμε όλες τις αντιστοιχες τιμές επικάλυψης. Αφότου αυτή η διαδικασία ολοκληρωθεί, στην συνέχεια εκτελείται ο μηχανισμός «Ενίσχυσης»[7] (Boosting). Πιο συγκεκριμένα, σε κάθε στήλη $i$ αντιστοιχεί και από μία τιμή ενίσχυσης $b_i \in \mathbb{R}_+$ η οποία πολλαπλασιάζεται με την τιμή επικάλυψής της. Οι τιμές αυτές ενημερώνονται κατά την φάση εκπαίδευσης του αλγορίθμου έτσι ώστε συχνά ενεργές στήλες να αντιστοιχούν σε χαμηλές τιμές ενίσχυσης ($0 < b_i < 1$), ενώ τυχόν στήλες οι οποίες ενεργοποιούνται σπανίως θα αντιστοιχούν σε τιμές ενίσχυσης $b_i > 1$. Μέσω του πολλαπλασιασμού των τιμών επικάλυψης $o_i$ με τις αντίστοιχες τιμές ενίσχυσης $b_i$, ο αλγόριθμος επιβάλλει μία σχετική ισορροπία όσο αφορά την συχνότητα ενεργοποίησης των στηλών, μέσω της οποίας εξάφαλίζεται η ιδιότητα της «κατανεμημένης» συμβολοσειράς, καθώς όπως θα δούμε στην συνέχεια οι τιμές της συμβολοσειράς εξόδου του αλγορίθμου καθορίζονται βάσει του συνόλου των ενεργών στηλών της περιοχής. Τέλος, λαμβάνοντας υπόψη ένα δεύτερο κατώφλι[8] $\theta_o$ ο αλγόριθμος θέτει $o_i = 0$ τις τιμές επικάλυψης εκείνες για τις οποίες ισχύει ότι $o_i \leq \theta_o$.

### 2. Ενεργοποίηση των στηλών

Μέσω του προηγούμενου βήματος έχουμε πλέον αποκομίσει ένα σύνολο τιμών επικάλυψης, μία για κάθε στήλη της περιοχής. Βάσει των τιμών αυτών ο αλγόριθμος καθορίζει ποιές από τις στήλες θα ενεργοποιηθούν και ποιες όχι, έτσι ώστε στην συνέχεια να κατασκευάσει την συμβολοσειρά εξόδου. Κάθε bit της εν λόγω συμβολοσειράς θα αντιστοιχεί και σε μία διαφορετική στήλη της περιοχής, ενώ η τιμή τους θα εξαρτάται από την κατάσταση της στήλης που τους αντιστοιχεί. Όσα bit αντιστοιχούν σε ενεργές στήλες θα λάβουν την τιμή «1», ενώ τα υπόλοιπα λαμβάνουν την τιμή «0».

Μία απλή και προφανής συνθήκη ενεργοποίησης των στηλών θα ήταν για παράδειγμα η σχέση $o_i > 0$, δηλαδή η ενεργοποίηση των στηλών στις οποίες αντιστοιχεί θετική τιμή επικάλυψης. Παρόλα αυτά θα θέλαμε η συμβολοσειρά εξόδου εκτός από «κατανεμημένη» να είναι και «αραιή». Για αυτό υπεύθυνος είναι ο μηχανισμός «Συστολής» (Inhibition). Ο μηχανισμός αυτός μπορεί να εκτελεστεί με δύο διαφορετικούς τρόπους[9], είτε ολικά λαμβάνοντας υπόψη όλες τις στήλες μαζί, είτε τοπικά θεωρώντας ότι κάθε στήλη ορίζει την δική της τοπική «γειτονιά». Σε κάθε περίπτωση απαραίτητη προϋπόθεση αποτελεί ο ορισμός μίας τιμής πυκνότητας[10] $d \in [0, 1]$ μέσω της οποίας εκφράζεται το επιθυμητό πλήθος ενεργών bit της συμβολοσειράς εξόδου. Παρακάτω περιγράφουμε τα δύο διαφορετικά είδη εκτέλεσης του μηχανισμού Συστολής.

- Ολική Συστολή: Βάσει των μεταβλητών που έχουμε ορίσει κατά την αρχικοποίηση, η θεμιτή πυκνότητα $d \in [0, 1]$ της εξόδου προσδιορίζεται μέσω του

[7] Βλέπε *Η Βιβλιοθήκη NuPIC / Η κλάση SpatialPooler / boostStrength.*

[8] Βλέπε *Η Βιβλιοθήκη NuPIC / Η κλάση SpatialPooler / stimulusThreshold.*

[9] Βλέπε *Η Βιβλιοθήκη NuPIC / Η κλάση SpatialPooler / globalInhibition.*

[10] Βλέπε *Η Βιβλιοθήκη NuPIC / Η κλάση SpatialPooler / localAreaDensity*, καθώς και *Η Βιβλιοθήκη NuPIC / Η κλάση SpatialPooler / numActiveColumnsPerInhArea.*

πλήθους $k = \text{int}(n \cdot d)$ των στηλών που θα θέλαμε να παραμείνουν ενεργοποιημένες, όπου $\text{int}(\cdot)$ η συνάρτηση μετατροπής ενός πραγματικού αριθμού σε ακέραιο, και $n$ το συνολικό πλήθος των στηλών της περιοχής. Ο αλγόριθμος ταξινομεί όλες τις στήλες της περιοχής ως προς την τιμή επικάλυψης, και ενεργοποιεί τις $k$ πρώτες στήλες στις οποίες αντιστοιχούν οι υψηλότερες από τις τιμές αυτές. Κατά αυτόν τον τρόπο εξασφαλίζουμε την ιδιότητα της «αραιής» συμβολοσειράς εισόδου, εφόσον φυσικά η τιμή $d$ είναι επαρκώς χαμηλή.

- Τοπική Συστολή: Σε αυτήν την περίπτωση η διαδικασία Συστολής εκτελείται ξεχωριστά για κάθε στήλη $i$ της περιοχής υπολογίζοντας ένα κατώφλι $k_i$ ως εξής[11]:

$$k_i = \text{int}\left(0.5 + n_i \cdot d\right)$$

όπου η τιμή $n_i$ ισούται με το μέγεθος της γειτονιάς της $i$-οστής στήλης[12]. Στην συνέχεια ο αλγόριθμος μετράει το πλήθος των γειτονικών στηλών, η τιμή επικάλυψης των οποίων είναι μεγαλύτερη από την αντίστοιχη τιμή επικάλυψης της στήλης προς εξέταση[13]. Εάν το εν λόγω πλήθος προκύψει χαμηλότερο της τιμής $k_i$, τότε η στήλη ενεργοποιείται. Η παραπάνω διαδικασία εκτελείται ξεχωριστά για κάθε στήλη της περιοχής.

Έχοντας υπολογίσει το σύνολο των ενεργοποιημένων στηλών ο αλγόριθμος μπορεί πλέον να κατασκευάσει την συμβολοσειρά εξόδου του αλγορίθμου, αντιστοιχίζοντας κάθε ενεργή στήλη στην τιμή «1», ενώ κάθε ανενεργή στήλη αντιστοιχείται στην τιμή «0».

### 3. Η Φάση Εκπαίδευσης του αλγορίθμου

Το τελευταίο βήμα αποβλέπει στην τροποποίηση των τιμών μονιμότητας των συνάψεων που αντιστοιχούν στα κεντρικά τμήματα δενδρίτη των στηλών της περιοχής, καθώς και στην ανανέωση των τιμών ορισμένων εσωτερικών μεταβλητών του αλγορίθμου, οι οποίες διέπουν την ορθή λειτουργία του. Επομένως είναι προφανές πως το βήμα αυτό θα πρέπει να εκτελείται μονάχα κατά την εκπαίδευση του συστήματος HTM, διαφορετικά το βήμα αυτό αγνοείται. Η φάση εκπαίδευσης του αλγορίθμου μπορεί να περιγραφεί συνοπτικά ως η εκτέλεση των παρακάτω πέντε βημάτων.

1. Ενημέρωση των τιμών μονιμότητας των συνάψεων: Κατά την αρχικοποίηση του αλγορίθμου ορίζουμε δύο παραμέτρους[14] $p_+$ και $p_- \in [0, 1]$, βάσει

[11] Γραμμή 1653 (`https://github.com/numenta/nupic/blob/master/src/nupic/algorithms/spatial_pooler.py`).

[12] Βλέπε *Η Βιβλιοθήκη NuPIC /Η κλάση SpatialPooler / inhibitionRadius.*

[13] Σε περίπτωση «ισοπαλίας» θα θεωρείται ότι μία γειτονική στήλη έχει υψηλότερη τιμή επικάλυψης εάν αυτή είναι ενεργοποιημένη, προκειμένου ότι έχει ήδη προηγηθεί η διαδικασία Συστολής στην γειτονιά της, γραμμές 1649-1651 (`https://github.com/numenta/nupic/blob/master/src/nupic/algorithms/spatial_pooler.py`).

[14] Βλέπε *Η Βιβλιοθήκη NuPIC / Η κλάση SpatialPooler / synPermActiveInc* και *Η Βιβλιοθήκη NuPIC / Η κλάση SpatialPooler / synPermInactiveDec.*

των οποίων τροποποιούνται οι τιμές μονιμότητας των συνάψεων των κεντρικών τμημάτων δενδρίτη. Σαρώνοντας τις συνάψεις που αντιστοιχούν στο τμήμα δενδρίτη κάθε ενεργής στήλης, ο αλγόριθμος αυξάνει τις τιμές μονιμότητας των ενεργών συνάψεων κατά $p_+$, ενώ αντίστοιχα μειώνει τις τιμές μονιμότητας των ανενεργών συνάψεων κατά $p_-$. Και στις δύο περιπτώσεις πραγματοποιείται έλεγχος μετά την τροποποίηση των συνάψεων, έτσι ώστε όλες οι ανανεωμένες τιμές μονιμότητας να βρίσκονται στο διάστημα $[0, 1]$.

2. Ενημέρωση των τιμών «overlapDutyCycle» και «activeDutyCycle»: Πρόκειται για δύο εσωτερικές μεταβλητές οι οποίες διατηρούνται ξεχωριστά για κάθε μία στήλη της περιοχής, και συμβάλλουν στην ενδυνάμωση των αδύναμων στηλών. Συγκεκριμένα:

   - $\text{overlapDutyCycle}_i$: Ένας κινητός μέσος όρος (sliding average) ο οποίος εκφράζει την συχνότητα βάσει της οποίας μία στήλη $i$ λαμβάνει τιμή επικάλυψης $o_i \geq \theta_o$. Η μεταβλητή αυτή χρησιμοποιείται στα πλαίσια εκτέλεσης του τρίτου βήματος τη φάσης εκπαίδευσης του αλγορίθμου.
   - $\text{overlapDutyCycle}_i$: Ένας κινητός μέσος όρος ο οποίος εκφράζει την συχνότητα ενεργοποίησης της στήλης $i$. Η μεταβλητή αυτή χρησιμοποιείται ως προς την ανανέωση των τιμών ενίσχυσης κάθε στήλης, η οποία πραγματοποιείται κατα το τέταρτο βήμα της φάσης εκπαίδευσης του αλγορίθμου.

3. Ενδυνάμωση των αδύναμων στηλών: Κατά την εκτέλεση του αλγορίθμου διατηρείται και από μία μεταβλητή «minOverlapDutyCycle» για κάθε στήλη $i$ της περιοχής, η τιμή της οποίας εκφράζει την ελάχιστη αποδεκτή τιμή overlapDutyCycle της στήλης που της αντιστοιχεί. Εάν για μία στήλη $i$ ισχυέι η σχέση $\text{overlapDutyCycle}_i < \text{minOverlapDutyCycle}_i$, τότε οι τιμές μονιμότητας κάθε σύναψης του κεντρικού τμήματος δενδρίτη που της αντιστοιχεί αυξάνονται κατά $p_+$.

4. Ενημέρωση των τιμών ενίσχυσης των στηλών: Οι τιμές ενίσχυσης $b_i$ κάθε στήλης $i$ της περιοχής ενημερώνονται βάσει της παρακάτω εξίσωσης:[15]

$$b_i = \exp\left[(d - \text{activeDutyCycle}_i) \cdot \text{boostStrength}\right]$$

   όπου $d$ η επιθυμητή πυκνότητα της συμβολοσειράς εξόδου[16], και boostStrength[17] μία θετική παράμετρος που ορίζεται κατά την αρχικοποίηση του αλγορίθμου, και η οποία εκφράζει το μέγεθος της συμβολής του μηχανισμού Ενίσχυσης.

[15] Γραμμή 1476 (`https://github.com/numenta/nupic/blob/master/src/nupic/algorithms/spatial_pooler.py`). Η εξίσωση αυτή αφορά την περίπτωση που εφαρμόζεται ολική, και όχι τοπική Συστολή. Σε διαφορετική περίπτωση η διαδικασία ανανέωσης των τιμών ενίσχυσης είναι ελαφρώς πιο περίπλοκη.

[16] Βλέπε *Η Βιβλιοθήκη NuPIC / Η κλάση SpatialPooler / localAreaDensity* καθώς και *Η Βιβλιοθήκη NuPIC / Η κλάση SpatialPooler / numActiveColumnsPerInhArea*.

[17] Βλέπε *Η Βιβλιοθήκη NuPIC / Η κλάση SpatialPooler / boostStrength*.

5. Έλεγχος έναρξης νέας περιόδου: Στα πλαίσια της εκτέλεσης του αλγορίθμου Spatial Pooler οι επαναλήψεις οργανώνονται σε περιόδους. Το πλήθος των επαναλήψεων που αποτελούν μία περίοδο ορίζεται κατά την αρχικοποίηση μέσω της παραμέτρου «dutyCyclePeriod»[18]. Στην αρχή κάθε νέας περιόδου πραγματοποιείται ενημέρωση της εσωτερικής μεταβλητής «inhibitionRadius» της περιοχής, καθώς και των τιμών «minOverlapDutyCycle» κάθε στήλης:

   - inhibitionRadius: Πρόκειται για μία τιμή βάσει της οποίας ορίζονται οι τοπικές «γειτονιές» των στηλών. Σε περίπτωση που εφαρμόζεται ολική Συστολή η τιμή αυτή ισούται πάντοτε με το πλήθος των στηλών της περιοχής, καθώς θεωρείται ότι όλες οι στήλες ανήκουν σε μία και μόνο ενιαία γειτονιά η οποία εκτείνεται σε ολόκληρη την περιοχή. Διαφορετικά η τιμή της μεταβλητής inhibitionRadius ενημερώνεται λαμβάνοντας υπόψη το μέσο πλήθος των συνδεδεμένων συνάψεων όλων των στηλών της περιοχής[19].
   - minOverlapDutyCycle: Σε κάθε στήλη $i$ αντιστοιχεί και από μία τιμή $\text{minOverlapDutyCycle}_i$ η οποία χρησιμοποιείται ως προς την ενδυνάμωση των αδύναμων στηλών στα πλαίσια της εκτέλεσης του τρίτου βήματος της φάσης εκπαίδευσης. Ο κανόνας ανανέωσης των τιμών τους είναι ο εξής:

     $$\text{minOverlapDutyCycle}_i = \text{minPctOverlapDutyCycle} \cdot M_i$$

     όπου $M_i$ η μέγιστη τιμή overlapDutyCycle που συναντάται ανάμεσα στις υπόλοιπες στήλες εντός της «γειτονιάς» της στήλης $i$. Σε περίπτωση που εφαρμόζεται ολική Συστολή η τιμή αυτή αντιστοιχεί στην μέγιστη τιμή overlapDutyCycle που συναντάται εντός ολόκληρης της περιοχής. Όσο αφορά την τιμή minPctOverlapDutyCycle, η μεταβλητή αυτή αποτελεί μία παράμετρο, η τιμή της οποίας καθορίζεται κατά την αρχικοποίηση του αλγορίθμου[20].

Εφόσον η φάση εκπαίδευσης ολοκληρωθεί, ο αλγόριθμος τερματίζει τροφοδοτώντας την συμβολοσειρά εξόδου ως είσοδο στον αλγόριθμο Temporal Pooler.

### 2.2.4 Ο Αλγόριθμος Temporal Pooler

Για να κατανοήσουμε την λειτουργία του αλγορίθμου Temporal Pooler θα πρέπει πρώτα να αναφερθούμε στις τρεις καταστάσεις στις οποίες μπορεί να βρίσκεται ένα κύτταρο ανά πάσα χρονική στιγμή. Συγκεκριμένα κάθε κύτταρο θα θεωρείται είτε «ενεργό» (active), είτε σε «κατάσταση πρόβλεψης» (predictive state), είτε «ανενεργό» (inactive). Θα πρέπει επιπλέον να τονίσουμε ότι ο αλγόριθμος Temporal Pooler εργάζεται πάνω στα κύτταρα της περιοχής, χωρίς να δίνει ιδιαίτερη σημασία

[18] Βλέπε *Η Βιβλιοθήκη NuPIC / Η κλάση SpatialPooler / dutyCyclePeriod.*

[19] Γραμμή 1036 (`https://github.com/numenta/nupic/blob/master/src/nupic/algorithms/spatial_pooler.py`).

[20] Βλέπε *Η Βιβλιοθήκη NuPIC / Η κλάση SpatialPooler / minPctOverlapDutyCycle.*

στις στήλες τις οποίες αυτά συνθέτουν. Όποτε λοιπόν χρησιμοποιούμε τον όρο «τμήματα δενδρίτη» στα πλαίσια της περιγραφής αυτού του αλγορίθμου, τότε θα αναφερόμαστε αποκλειστικά και μόνο στα «περιφερικά τμήματα δενδρίτη» των κυττάρων, οι συνάψεις των οποίων ωστόσο θυμίζουν εκείνες των κεντρικών τμημάτων δενδρίτη, υπό την έννοια ότι και αυτές χαρακτηρίζονται από τις ίδιες τρεις ιδιότητες της «ενεργής», της «συνδεδεμένης» και της «ενεργοποιημένης» σύναψης. Αναφορικά με τα περιφερικά τμήματα δενδρίτη ενδέχεται και αυτά με την σειρά τους να κατέχουν τις παρακάτω ιδιότητες:

- Συμβατό: Ένα τμήμα δενδρίτη του οποίου το πλήθος των ενεργών[21] συνάψεων –είτε συνδεδεμένων[22] είτε μη– ξεπερνάει την τιμή ενός κατωφλιού minThreshold[23] το οποίο ορίζεται κατά την αρχικοποίηση του αλγορίθμου.
- Ενεργό: Ένα τμήμα δενδρίτη του οποίου το πλήθος των ενεργοποιημένων[24] συνάψεων ξεπερνάει την τιμή ενός κατωφλιού activationTheshold[25] το οποίο ορίζεται κατά την αρχικοποίηση του αλγορίθμου.

Ενώ τα κύτταρα μίας περιοχής θα πρέπει κάθε χρονική στιγμή να βρίσκονται σε μία και μόνο από τις τρεις καταστάσεις που αναφέραμε, τα τμήματα δενδρίτη από την άλλη είναι εφικτό να θεωρούνται ενεργά και συμβατά ταυτόχρονα, ή ακόμη και τίποτα από τα δύο. Έχοντας επομένως αναφέρει τα παραπάνω μπορούμε πλέον να προχωρήσουμε στην περιγραφή της λειτουργίας του αλγορίθμου Temporal Pooler, η οποία στην ουσία σχετίζεται με τον καθορισμό της κατάστασης κάθε κυττάρου μίας περιοχής, και μπορεί να συνοψιστεί σε δύο επιμέρους βήματα:

1. Υπολογισμός των ενεργών κυττάρων.
2. Υπολογισμός των κυττάρων που θα εισέλθουν σε κατάσταση πρόβλεψης.

Σε αντίθεση με τον αλγόριθμο Spatial Pooler, ο αλγόριθμος αυτός δεν εμπεριέχει κάποιο ξεχωριστό βήμα εκπαίδευσης. Παρόλα αυτά στην περίπτωση που το μοντέλο βρίσκεται σε λειτουργία εκμάθησης πραγματοποιούνται παράλληλα ορισμένες διαδικασίες εκπαίδευσης σε κάθε ένα από τα δύο επιμέρους βήματα.

Γενικά θα λέγαμε ότι το πρώτο βήμα του αλγορίθμου αποσκοπεί κυρίως στην κατασκευή της συμβολοσειράς εξόδου, η οποία παράλληλα αποτελεί και την έξοδο ολόκληρης της περιοχής. Πρόκειται για μία δυαδική συμβολοσειρά αρκετά μεγάλου μήκους, καθώς κάθε bit αντιστοιχεί και σε ένα κύτταρο της περιοχής, ενώ η τιμή του εξαρτάται από την κατάσταση του κυττάρου. Συγκεκριμένα εάν ένα κύτταρο είναι ενεργό τότε το αντίστοιχο bit λαμβάνει την τιμή «1», αλλιώς λαμβάνει την τιμή

[21] Ως ενεργή σύναψη ενός κυττάρου περιγράφουμε την σύναψη η οποία το συνδέει με ένα άλλο ενεργό κύτταρο εντός της ίδιας περιοχής.

[22] Ως συνδεδμένη σύναψη σύναψη ενός κυττάρου περιγράφουμε την σύναψη της οποίας η τιμή μονιμότητας ξεπερνάει την τιμή κατωφλιού connectedPermanence, η οποία ορίζεται κατά την αρχικοποίηση του αλγορίθμου.

[23] Βλέπε *Η Βιβλιοθήκη NuPIC / Η Κλάση TemporalMemory / minThreshold.*

[24] Όπως και με την περίπτωση των συνάψεων που ανήκουν σε κεντρικά τμήματα δενδρίτη, μία σύναψη θεωρείται ενεργοποιημένη εάν και μόνο εάν η σύναψη είναι ενεργή και συνδεδεμένη.

[25] Βλέπε *Η Βιβλιοθήκη NuPIC / Η Κλάση TemporalMemory / activationThreshold.*

«0». Λόγω του ότι ένα κύτταρο προκειμένου να ενεργοποιηθεί θα πρέπει υποχρεωτικά να ανήκει σε μία ενεργή στήλη, ο αλγόριθμος εγγυάται ότι η συμβολοσειρά εξόδου θα αποτελεί επίσης συμβολοσειρά SDR.

Το δεύτερο βήμα του αλγορίθμου από την άλλη επιδρά έμμεσα στην συμβολοσειρά εξόδου. Όπως θα δούμε παρακάτω ορισμένα κύτταρα τα οποία βρίσκονται σε κατάσταση πρόβλεψης κατά την επανάληψη $t$ ενδέχεται να ενεργοποιηθούν κατά την επόμενη επανάληψη $t+1$. Μέσω των κυττάρων αυτών το σύστημα στην ουσία εκτελεί προβλέψεις, καθώς εισάγοντας ένα κύτταρο σε κατάσταση πρόβλεψης την χρονική στιγμή $t$, το σύστημα προβλέπει την ενεργοποίηση της στήλης στην οποία το κύτταρο αυτό ανήκει για την αμέσως επόμενη χρονικη στιγμή $t+1$. Κύτταρα τα οποία εκτελούν σωστές προβλέψεις επιβραβεύονται, ενώ οι λανθασμένες προβλέψεις τιμωρούνται. Κατά αυτον τον τρόπο το σύστημα εκπαιδεύεται ως προς την ανίχνευση των χρονικών μοτίβων που παρουσιάζουν τα δεδομένα, λαμβάνοντας υπόψη τις εξαρτήσεις μεταξύ των δεδομένων. Παρακάτω περιγράφουμε λεπτομερέστερα την εκτέλεση των δύο αυτών βημάτων του αλγορίθμου.

### 1. Υπολογισμός των ενεργών κυττάρων

Ο αλγόριθμος Temporal Pooler ξεκινάει λαμβάνοντας την έξοδο του αλγορίθμου Spatial Pooler, δηλαδή την δυαδική συμβολοσειρά που υποδεικνύει το σύνολο των ενεργών στηλών της περιοχής. Βάσει αυτής ο αλγόριθμος σαρώνει κάθε ενεργή στήλη έτσι ώστε να προσδιορίσει τα κύτταρα που πρόκειται να ενεργοποιηθούν. Απαραίτητη λοιπόν προϋπόθεση ενεργοποίησης ενός κυττάρου, είναι το κύτταρο αυτό να ανήκει σε μία ενεργή στήλη. Επιπλέον εάν το μοντέλο βρίσκεται σε λειτουργία εκμάθησης, τότε σαρώνονται και οι υπόλοιπες ανενεργές στήλες με σκοπό την εφαρμογή ορισμένων διαδικασιών εκπαίδευσης. Σαρώνοντας κάθε στήλη της περιοχής ο αλγόριθμος επομένως εξετάζει σε ποιά από τις τρεις παρακάτω περιπτώσεις αυτή ανήκει, και έπειτα πράττει αναλόγως:

1. Εάν μία στήλη είναι ενεργή και περιέχει τουλάχιστον ένα κύτταρο σε κατάσταση πρόβλεψης[26], τότε τα κύτταρα αυτά και μόνο ενεργοποιούνται, καθώς επίσης σημειώνονται ως «κύτταρα-νικητές»[27]. Επιπλέον εάν το μοντέλο βρίσκεται σε λειτουργία εκμάθησης, τότε ο αλγόριθμος σαρώνει κάθε ενεργό τμήμα δενδρίτη[28,29] κάθε νικητήριου κυττάρου, και εκτελεί τα εξής βήματα[30]:

[26] Ορισμένα κύτταρα ενδέχεται να βρίσκονται σε κατάσταση πρόβλεψης βάσει της αμέσως προηγούμενης επανάληψης, βλέπε *Ο Αλγόριθμος Temporal Pooler / 2. Υπολογισμός των κυττάρων που θα εισέλθουν σε κατάσταση πρόβλεψης*.

[27] Εκτός από τις τρεις βασικές καταστάσεις των κυττάρων (ενεργή, πρόβλεψης και ανενεργή) ο αλγόριθμος σημειώνει επιπλέον ορισμένα κύτταρα ως «κύτταρα-νικητές». Η διαδικασία αυτή συνδέεται με την λειτουργία εκμάθησης.

[28] Βλέπε *Ο Αλγόριθμος Temporal Pooler / 2. Υπολογισμός των κυττάρων που θα εισέλθουν σε κατάσταση πρόβλεψης*, καθώς και *Η Βιβλιοθήκη NuPIC / Η Κλάση TemporalMemory / activationThreshold*.

[29] Τα κύτταρα-νικητές είναι βέβαιο πως θα περιέχουν τουλάχιστον ένα ενεργό τμήμα δενδρίτη βάσει της αμέσως προηγούμενης επανάληψης, εφόσον τα κύτταρα αυτά πριν την ενεργοποίησή τους βρίσκονται σε κατάσταση πρόβλεψης, βλέπε *Ο Αλγόριθμος Temporal Pooler / 2. Υπολογισμός των κυττάρων που θα εισέλθουν σε κατάσταση πρόβλεψης*.

[30] Γραμμή 448 (`https://github.com/numenta/nupic/blob/master/src/nupic/algorithms/`

(α′) Αύξηση της τιμής μονιμότητας των ενεργών συνάψεων του τμήματος δενδρίτη[31].

(β′) Μείωση της τιμής μονιμότητας των ανενεργών συνάψεων του τμήματος δενδρίτη[32].

(γ′) Τυχαία δημιουργία νέων συνάψεων μεταξύ του τμήματος δενδρίτη και των κυττάρων-νικητών της αμέσως προηγούμενης επανάληψης, εάν τέτοιες συνάψεις δεν υπάρχουν ήδη. Το πλήθος αυτών των συνάψεων ισούται με το μέγιστο πλήθος των νέων συνάψεων που μπορούν να δημιουργηθούν κατά την διάρκεια της εκπαίδευσης[33] μείον το πλήθος των τωρινών ενεργών συνάψεων του τμήματος δενδρίτη προς εξέταση, αρκεί το πλήθος αυτό να μην υπερβαίνει ένα συγκεκριμένο κατώφλι.[34].

2. Στην περίπτωση που μία στήλη ναι μεν είναι ενεργή αλλά δεν περιέχει κανένα κύτταρο σε κατάσταση πρόβλεψης, τότε πραγματοποιείται ο μηχανισμός «Έκρηξης»[35] (Bursting) ο οποίος αφορά την εκτέλεση των εξής δύο βημάτων[36]:

   (α′) Ενεργοποίηση κάθε κυττάρου της στήλης.

   (β′) Επιλογή ενός και μόνο κυττάρου-νικητή: Σε αντίθεση με την προηγούμενη περίπτωση κατά την οποία κάθε κύτταρο που ενεργοποιείται σημειώνεται επιπλέον ως κύτταρο-νικητής, στα πλαίσια του μηχανισμού Έκρηξης μονάχα ένα από τα κύτταρα της στήλης αναδεικνύεται ως «νικητής». Η διαδικασία έυρεσης του νικητή αφορά την εξέταση κάθε τμήματος δενδρίτη κάθε κυττάρου της στήλης:

      - Εάν υπάρχει έστω και ένα συμβατό[37] τμήμα δενδρίτη τότε ο αλγόριθμος βρίσκει το πιο συμβατό απο αυτά –δηλαδή το τμήμα δενδρίτη με τις περισσότερες ενεργές συνάψεις– και σημειώνει το κύτταρο στο οποίο ανήκει ως «νικητή». Επιπλέον σε περίπτωση που το μοντέλο βρίσκεται σε λειτουργία εκμάθησης ο αλγόριθμος προχωρά σε ορισμένες ενέργειες που σχετίζονται με τις συνάψεις του συγκεκριμένου τμήματος δενδρίτη. Οι ενέργειες αυτές ταυτίζονται με τις αντίστοιχες τροποποιήσεις που εκτελούνται κατά τα βήματα α′, β′ και γ′ της περίπτωσης 1.

---

`temporal_memory.py`).

[31] Βλέπε *Η Βιβλιοθήκη NuPIC / Η Κλάση TemporalMemory / permanenceIncrement.*

[32] Βλέπε *Η Βιβλιοθήκη NuPIC / Η Κλάση TemporalMemory / permanenceDecrement.*

[33] Η τιμή αυτή καθορίζεται κατά την αρχικοποίηση του αλγορίθμου, βλέπε *Η Βιβλιοθήκη NuPIC / Η Κλάση TemporalMemory / maxNewSynapseCount.*

[34] Η τιμή του κατωφλιού καθορίζεται κατά την αρχικοποίηση του αλγορίθμου και αφορά το μέγιστο πλήθος συνάψεων που μπορεί να αντιστοιχεί σε ένα περιφερικό τμήμα δενδρίτη, βλέπε *Η Βιβλιοθήκη NuPIC / Η Κλάση TemporalMemory / maxSynapsesPerSegment.*

[35] Να σημειωθεί ότι κατά το πέρασμα του πρώτου στοιχείου μίας ακολουθίας πραγματοποιείται πάντοτε ο μηχανισμός Έκρηξης, καθώς κατά το ξεκίνημα της ακολουθίας κάθε κύτταρο θεωρείται ανενεργό, δηλαδή δεν υπάρχει περίπτωση κάποιο κύτταρο να βρίσκεται σε κατάσταση πρόβλεψης.

[36] Γραμμή 529 (`https://github.com/numenta/nupic/blob/master/src/nupic/algorithms/temporal_memory.py`).

[37] Η ύπαρξη ή μη των συμβατών τμημάτων δενδρίτη των κυττάρων έχει ήδη εξακριβωθεί κατά το δεύτερο βήμα του αλγορίθμου Temporal Pooling της αμέσως προηγούμενης επανάληψης, βλέπε *Ο Αλγόριθμος Temporal Pooler / 2. Υπολογισμός των κυττάρων που θα εισέλθουν σε κατάσταση πρόβλεψης.*

- Σε περίπτωση που δεν υπάρχει κανένα συμβατό τμήμα δενδρίτη μεταξύ των κυττάρων της στήλης τότε ο αλγόριθμος σημειώνει ως «κύτταρο-νικητή» το κύτταρο στο οποίο αντιστοιχεί το μικρότερο πλήθος τμημάτων δενδρίτη. Επιπλέον σε περίπτωση που το μοντέλο βρίσκεται σε λειτουργία εκμάθησης τότε ο αλγόριθμος προσθέτει ένα νέο τμήμα δενδρίτη στο κύτταρο αυτό, το οποίο (το τμήμα δενδρίτη) περιέχει ένα σύνολο συνάψεων[38]. Κάθε μία από αυτές τις συνάψεις συνδέει το εκάστοτε κύτταρο-νικητή και με ένα κύτταρο-νικητή της αμέσως προηγούμενης επανάληψης.

3. Τέλος εάν το μοντέλο βρίσκεται σε λειτουργία εκμάθησης και μία στήλη είναι ανενεργή, τότε ο αλγόριθμος «τιμωρεί» τα συμβατά τμήματα δενδρίτη –εάν αυτά υπάρχουν– που αντιστοιχούν στα κύτταρά της, καθώς κάτι τέτοιο θα σήμαινε ότι η πρόβλεψή τους ήταν λανθασμένη, γεγονός το οποίο επιβεβαιώνεται εφόσον η στήλη προς εξέταση πράγματι δεν ενεργοποιήθηκε[39]. Για κάθε ένα από αυτά τα τμήματα δενδρίτη ο αλγόριθμος μειώνει την τιμή μονιμότητας των συνάψεών του κατά μία τιμή predictedSegmentDecrement[40] η οποία ορίζεται κατά την αρχικοποίηση του αλγοριθμου. Να σημειωθεί ότι στην περίπτωση που η τιμή αυτής της παραμέτρου έχει οριστεί ίση με το μηδέν, τότε η διαδικασία αυτή παραβλέπεται.

### 2. Υπολογισμός των κυττάρων που θα εισέλθουν σε κατάσταση πρόβλεψης

Το δεύτερο και τελευταίο βήμα του αλγορίθμου αποτελείται από δύο επιμέρους βήματα[41] τα οποία βασικά αφορούν τα τμήματα δενδρίτη κάθε κυττάρου της περιοχής, εφόσον μέσω αυτών καθορίζεται το σύνολο των κυττάρων που πρόκειται να εισέλθουν σε κατάσταση πρόβλεψης:

1. Υπολογισμός των ενεργών τμημάτων δενδρίτη: Σε αυτό το βήμα ο αλγόριθμος σαρώνει κάθε τμήμα δενδρίτη κάθε κυττάρου της περιοχής, και βάσει μίας τιμής κατωφλιού[42] κρίνει ποιά από αυτά θα ενεργοποιηθούν και ποιά

[38] Εάν το πλήθος αυτών των νέων συνάψεων ξεπερνάει την τιμή της παραμέτρου maxNewSynapseCount, τότε η επιλογή γίνεται κατά τυχαίο τρόπο μέχρι να συμπληρωθούν έως και maxNewSynapseCount συνάψεις, βλέπε *Η Βιβλιοθήκη NuPIC / Η Κλάση TemporalMemory / maxNewSynapseCount*.

[39] Για να είμαστε βέβαια πιο ακριβείς, η συμβατότητα ενός τμήματος δενδρίτη δεν συνεπάγεται και την ενεργοποίησή του, καθώς οι ενεργές συνάψεις του ενδέχεται να είναι μη-συνδεδεμένες. Αυτό σημαίνει ότι ένα κύτταρο στο οποίο αντιστοιχούν συμβατά τμήματα δενδρίτη, αν και πιθανό, δεν είναι απολύτως βέβαιο ότι θα βρίσκεται σε κατάσταση πρόβλεψης –όπως θα δούμε στο επόμενο βήμα του αλγορίθμου τα κύτταρα στα οποία αντιστοιχεί τουλάχιστον ένα ενεργό τμήμα δενδρίτη εισέρχονται σε κατάσταση πρόβλεψης. Παρόλα αυτά στην περίπτωση που οι ενεργές συνάψεις τύχαινε να ήταν ταυτόχρονα και συνδεδεμένες, τότε τα συμβατά τμήματα δενδρίτη θα θεωρούνταν ενεργά με αποτέλεσμα την είσοδο του κυττάρου σε κατάσταση πρόβλεψης. Η πρόβλεψη σε αυτήν την περίπτωση θα ήταν σαφώς λανθασμένη από την στιγμή που το κύτταρο ανήκει σε μία ανενεργή στήλη.

[40] Βλέπε *Η Βιβλιοθήκη NuPIC / Η Κλάση TemporalMemory / predictedSegmentDecrement*.

[41] Γραμμή 264 (`https://github.com/numenta/nupic/blob/master/src/nupic/algorithms/temporal_memory.py`).

[42] Βλέπε *Η Βιβλιοθήκη NuPIC / Η Κλάση TemporalMemory / activationThreshold*.

όχι. Στην συνέχεια κάθε κύτταρο το οποίο περιέχει τουλάχιστον ένα ενεργό τμήμα δενδρίτη εισέρχεται σε κατάσταση πρόβλεψης.

2. Υπολογισμός των συμβατών τμημάτων δενδρίτη: Τέλος ο αλγόριθμος υπολογίζει το σύνολο των τμημάτων δενδρίτη που θεωρούνται συμβατά[43].

Μολονότι οι διαδικασίες που εκτελούνται σε αυτό το βήμα δεν επιδρούν άμεσα πάνω στην τωρινή έξοδο, καθορίζουν ωστόσο την έξοδο της επόμενης επανάληψης του αλγορίθμου καθώς και την εκπαίδευση του μοντέλου.

### 2.2.5 Η Βιβλιοθήκη NuPIC

Με σκοπό την ορθή κατασκευή συστημάτων HTM η εταιρεία «Numenta» προσφέρει την βιβλιοθήκη ανοιχτού κώδικα «NuPIC»[44] –συμβατή με την γλώσσα προγραμματισμού Python 2.7– η οποία εμπεριέχει όλα τα απαραίτητα εργαλεία ως προς την εκτέλεση των αλγορίθμων που έχουμε περιγράψει. Παρακάτω αναφέρουμε τις τέσσερεις κλάσεις που πρόκειται να χρησιμοποιήσουμε στα πλαίσια της εκτέλεσης των πειραμάτων αυτής της εργασίας, εστιάζοντας στην σημασία των παραμέτρων της καθεμίας.

#### Η Κλάση ScalarEncoder[45]

Μέσω της κλάσης αυτής ορίζεται η συνάρτηση κωδικοποίησης όπως την έχουμε περιγράψει στο τμήμα της εργασίας με τίτλο *Οι Αραιές Κατανεμημένες Αναπαραστάσεις*. Η κλάση αυτή αφορά την επιτυχή κωδικοποίηση αριθμητικών χαρακτηριστικών ως δυαδικές συμβολοσειρές. Παρακάτω παρουσιάζουμε τις παραμέτρους του κατασκευαστή της κλάσης.

- w (int): Το επιθυμητό πλήθος των ενεργών bit της κωδικοποιημένης συμβολοσειράς. Η τιμή αυτής της παραμέτρου θα πρέπει πάντοτε να είναι περιττή.
- minval (float): Το κάτω όριο του εύρους τιμών του χαρακτηριστικού προς κωδικοποίηση.
- maxval (float): Το άνω όριο του εύρους τιμών του χαρακτηριστικού προς κωδικοποίηση.
- periodic (bool): Εάν η παράμετρος αυτή έχει την τιμή «Αληθής» τότε το εύρος τιμών $[\text{minval}, \text{maxval}]$ τυλίγεται γύρω από τον εαυτό του έτσι ώστε τιμές $x$ για τις οποίες ισχύει $x > \text{maxval}$ να θεωρούνται από την συνάρτηση ως $x' = \text{minval} + (x - \text{maxval})$. Το ίδιο ισχύει και για τις τιμές $x < \text{minval}$.
- n (int): Το συνολικό επιθυμητό πλήθος των bit της κωδικοποιημένης συμβολοσειράς.

[43] Βλέπε *Η Βιβλιοθήκη NuPIC / Η Κλάση TemporalMemory / minThreshold*.
[44] Έκδοση 1.0.5.: `https://nupic.docs.numenta.org/1.0.5/index.html`
[45] nupic.encoders.scalar.ScalarEncoder

- radius (int): Η παράμετρος αυτή προσφέρει έναν εναλλακτικό τρόπο καθορισμού του συνολικού πλήθους bit της κωδικοποιημένης συμβολοσειράς. Θέτοντας την τιμή αυτή ορίζουμε ένα μήκος συμβολοσειράς $n$, τέτοιο ώστε οι συμβολοσειρές που προκύπτουν από την κωδικοποίηση δύο διαφορετικών τιμών εισόδου $x_1, x_2$, όπου $\mid x_1 - x_2 \mid \geq$ radius, να μην μοιράζονται κανένα ενεργό bit, δηλαδή οι τιμές $x_1, x_2$ θα αντιστοιχούν σε διαφορετικούς και απομακρυσμένους κάδους.

- resolution (int): Η παράμετρος αυτή προσφέρει έναν δεύτερο εναλλακτικό τρόπο καθορισμού του συνολικού πλήθους bit της κωδικοποιημένης συμβολοσειράς. Θέτοντας την τιμή αυτή ορίζουμε ένα μήκος συμβολοσειράς $n$, το οποίο μας εγγυάται ότι οι συμβολοσειρές που προκύπτουν από την κωδικοποίηση δύο διαφορετικών τιμών εισόδου $x_1, x_2$, όπου $\mid x_1 - x_2 \mid \geq$ resolution, θα διαφέρουν μεταξύ τους ακόμη και εάν μοιράζονται μερικά ενεργά bit, δηλαδή οι τιμές $x_1, x_2$ θα αντιστοιχούν ναι μεν σε διαφορετικούς αλλά γειτονικούς «κάδους».

- name (str): Μέσω αυτής της παραμέτρου αποδίδουμε ένα όνομα σε ένα αντικείμενο της κλάσης με σκοπό τον εύκολο διαχωρισμό πολλαπλών αντικειμένων της ίδιας κλάσης.

- clipInput (bool): Εάν η παράμετρος αυτή έχει την τιμή «Αληθής» τότε τιμές οι οποίες βρίσκονται εκτός του διαστήματος [minval, maxval] κωδικοποιούνται θεωρούμενες είτε ως minval είτε ως maxval, αναλόγως της τιμής του άκρου από το οποίο απέχουν μικρότερη απόσταση.

- forced (bool): Εάν η παράμετρος αυτή έχει την τιμή «Αληθής» τότε αγνοούνται μερικοί εσωτερικοί έλεγχοι που εκτελεί ο κατασκευαστής. Για παράδειγμα σε περίπτωση που η τιμή της παραμέτρου είναι «Ψευδής» τότε θα πρέπει απαραίτητα να ισχύει $w \geq 21$.

### Η Κλάση CategoryEncoder[46]

Η κλάση CategoryEncoder χρησιμοποιείται ως προς την κωδικοποίηση κατηγορικών χαρακτηριστικών, οι τιμές των οποίων δεν εκφράζουν κάποια ποσότητα. Παρακάτω παρουσιάζουμε τις παραμέτρους του κατασκευαστή της κλάσης.

- w (int): Το επιθυμητό πλήθος των ενεργών bit της κωδικοποιημένης συμβολοσειράς. Η τιμή αυτής της παραμέτρου θα πρέπει πάντοτε να είναι περιττή.
- categoryList (list): Μία λίστα η οποία περιέχει όλες τις διακριτές τιμές του χαρακτηριστικού προς κωδικοποίηση. Δεδομένης μίας λίστα μήκους $L$, το συνολικό πλήθος των bit των κωδικοποιημένων συμβολοσειρών θα ισούται με $(L + 1) \cdot \mathrm{w}$. Η τιμή $L + 1$ στο γεγονός ότι η κλάση ορίζει πάντοτε μία επιπλέον τιμή «0» βάσει της οποίας κωδικοποιούνται τυχόν άγνωστες τιμές, δηλαδή τιμές οι οποίες δεν ορίστηκαν μέσω της παραμέτρου categoryList κατά την αρχικοποίηση του αντικειμένου[47]. Θα πρέπει να τονίσουμε ότι δύο κωδικοποιημένες συμβολοσειρές που παράγονται βάσει δύο διακριτών τιμών $x_1, x_2$, όπου $x_1 \neq x_2$, δεν μοιράζονται κανένα ενεργό bit μεταξύ τους.
- name (str): Μέσω αυτής της παραμέτρου αποδίδουμε ένα όνομα σε ένα αντικείμενο της κλάσης με σκοπό τον εύκολο διαχωρισμό πολλαπλών αντικειμένων της ίδιας κλάσης.
- forced (bool): Εάν η παράμετρος αυτή έχει την τιμή «Αληθής» τότε αγνοούνται μερικοί εσωτερικοί έλεγχοι που εκτελεί ο κατασκευαστής. Για παράδειγμα σε περίπτωση που η τιμή της παραμέτρου είναι «Ψευδής» τότε θα πρέπει απαραίτητα να ισχύει $\mathrm{w} \geq 21$.

Πριν προχωρήσουμε στην περιγραφή της κλάσης «SpatialPooler» θα πρέπει να αναφέρουμε ότι σε κάθε μία από τις δύο παραπάνω κλάσεις ScalarEnocder και CategoryEncoder ανήκει μία συνάρτηση *encode*, μέσω της οποίας εκτελείται η κωδικοποίηση των τιμών καθε χαρακτηριστικού. Σε κάθε περίπτωση η συνάρτηση δέχεται την τιμή προς κωδικοποίηση ως την μοναδική παράμετρο *inputData*, και επιστρέφει την δυαδική της κωδικοποίηση υπό την μορφή ενός μονοδιάστατου NumPy πίνακα.

### Η Κλάση SpatialPooler[48]

Η κλάση αυτή είναι υπεύθυνη για την αρχικοποίηση και την εκτέλεση του αλγορίθμου Spatial Pooler, ο οποίος περιγράφεται στο τμήμα αυτής της εργασίας με τίτλο *Ο Αλγόριθμος Spatial Pooler*. Στις δύο επόμενες σελίδες παρουσιάζουμε τις παραμέτρους του κατασκευαστή της κλάσης.

[46] nupic.encoders.category.CategoryEncoder

[47] Σε περίπτωση που θέλουμε να κατασκευάσουμε συμβολοσειρές μήκους $L \cdot \mathrm{w}$ μπορούμε να παραλείψουμε την προσθήκη μίας πιθανής τιμής του χαρακτηριστικού προς κωδικοποίηση.

[48] nupic.algorithms.spatial_pooler.SpatialPooler

- inputDimensions (int ή tuple): Η διαστασημότητα της εισόδου. Ορίζοντας την παράμετρο αυτή ως μία πλειάδα $n > 1$ τιμών, εφαρμόζουμε μία συγκεκριμένη $n$-διάστατη τοπολογία στον τρόπο οργάνωσης της εισόδου του αλγορίθμου.

- columnDimensions (int ή tuple): Το πλήθος των στηλών μίας περιοχής, και ως αποτέλεσμα αυτού, το μήκος της συμβολοσειράς εξόδου του αλγορίθμου. Ορίζοντας την παράμετρο αυτή ως μία πλειάδα $n > 1$ τιμών, εφαρμόζουμε μία συγκεκριμένη $n$-διάστατη τοπολογία στον τρόπο οργάνωσης των στηλών.

- potentialRadius (int): Η τιμή αυτής της παραμέτρου συμβάλλει στον καθορισμό του πλήθους των bit εισόδου με τα οποία κάθε στήλη της περιοχής ενδέχεται να είναι συνεδεμένη. Στην περίπτωση που δεν εφαρμόζεται κάποια συγκεκριμένη τοπολογία στην είσοδο –δηλαδή κατέχει μονοδιάστατη τοπολογία– τότε η τιμή της παραμέτρου ορίζει άμεσα το παραπάνω πλήθος.

- potentialPct (float $\in [0, 1]$): Η τιμή αυτή ορίζει το ποσοστό των bit εισόδου εντός του potentialRadius μίας στήλης με τα οποία η στήλη ενδέχεται να είναι συνδεδεμένη. Η παράμετρος αυτή χρησιμοποιείται σε περίπτωση που η τιμή potentialRadius των στηλών είναι υπερβολικά υψηλή, με αποτέλεσμα να υπάρχει μεγάλο σύνολο αλληλοεπικαλυπτόμενων bit εισόδου. Για παράδειγμα εάν το potentialRadius μίας στήλης ισούται με 100, ενώ το potentialPct ισούται με 0.9, τότε η στήλη ενδέχεται να είναι συνδεδεμένη με $100 \cdot 0.1 = 90$ bit εισόδου.

- globalInhibiton (bool): Εάν η παράμετρος αυτή έχει την τιμή «Αληθής» τότε ο αλγόριθμος εφαρμόζει τον μηχανισμό ολικής Συστολής κατά το δεύτερο βήμα εκτέλεσής του, διαφορετικά εφαρμόζεται τοπική Συστολή. Γενικά προτείνεται η τιμή «Αληθής» της παραμέτρου καθώς αυτό οδηγεί σε σημαντική μείωση του χρόνου εκτέλεσης του αλγορίθμου, καθώς επίσης και σε ευκολότερο έλεγχο της τιμής πυκνότητας της συμβολοσειράς εξόδου.

- localAreaDensity (float $\in [0, 1]$): Σε περίπτωση εφαρμογής του μηχανισμού ολικής Συστολής η τιμή αυτή εκφράζει το επιθυμητό ποσοστό ενεργοποίησης των στηλών της περιοχής, με αποτέλεσμα τον εύκολο καθορισμό της τιμής πυκνότητας της συμβολοσειράς εξόδου. Διαφορετικά η τιμή αυτή συμβάλλει έμμεσα στην ενεργοποίηση μίας στήλης κατά την εκτέλεση της τοπικής Συστολής. Σε περίπτωση χρήσης αυτής της παραμέτρου για τον καθορισμό των ενεργοποιημένων στηλών θα πρέπει οπωσδήποτε να ισχύει numActiveColumnsPerInhArea $< 0$.

- numActiveColumnsPerInhArea (int): Η τιμή αυτή εκφράζει το μέγιστο πλήθος των στηλών που παραμένουν ενεργοποιημένες εντός μίας τοπικής «γειτονιάς» μίας στήλης. Σε περίπτωση που εφαρμόζεται ο μηχανισμός ολικής Συστολής η τιμή αυτή αφορά ολόκληρο το σύνολο στηλών της περιοχής, με αποτέλεσμα τον εύκολο καθορισμό της τιμής πυκνότητας της συμβολοσειράς εξόδου του αλγορίθμου. Σε περίπτωση χρήσης αυτής της παραμέτρου

για τον καθορισμό των ενεργοποιημένων στηλών, θα πρέπει οπωσδήποτε να ισχύει localAreaDensity $< 0$.

- stimulusThreshold (int): Η τιμή αυτή αποτελεί ένα κατώφλι για την ενδεχόμενη ενεργοποίηση κάθε στήλης. Σε περίπτωση που η τιμή επικάλυψης $o_i$ μίας στήλης $i$ πολλαπλασιασμένη με την αντίστοιχη τιμή ενίσχυσης $b_i$ είναι χαμηλότερη της τιμής stimulusThreshold, τότε ορίζεται $o_i = 0$, γεγονός το οποίο σημαίνει πως η στήλη δεν πρόκειται να ενεργοποιηθεί.

- synPermInactiveDec (float $\in [0, 1]$): Τιμή βάσει της οποίας μειώνεται η τιμή μονιμότητας των ανενεργών συνάψεων κατά την φάση εκπαίδευσης του αλγορίθμου.

- synPermActiveInc (float $\in [0, 1]$): Τιμή βάσει της οποίας αυξάνεται η τιμή μονιμότητας των ενεργών συνάψεων κατά την φάση εκπαίδευσης του αλγορίθμου.

- synPermConnected (float $\in [0, 1]$): Μία τιμή κατωφλιού η οποία σε συνδυασμό με την τιμή μονιμότητας μίας σύναψης καθορίζει εάν αυτή θεωρείται συνδεδεμένη ή όχι. Επιπλέον κατά την αρχικοποίηση οι τιμές μονιμότητας κάθε σύναψης λαμβάνουν τυχαίες τιμές γύρω από αυτήν την τιμή.

- minPctOverlapDutyCycle (float $\in [0, 1]$): Η τιμή αυτή χρησιμοποιείται για τον υπολογισμό των τιμών minOverlapDutyCycle των στηλών, οι οποίες εκφράζουν την ελάχιστη αποδεκτή τιμή overlapDutyCycle που μπορούν να λάβουν ούτως ώστε να μην θεωρηθούν «αδύναμες».

- dutyCyclePeriod (int): Το πλήθος των επαναλήψεων που ορίζουν μία «περίοδο». Στην αρχή κάθε νέας περιόδου πραγματοποιείται ανανέωση των τιμών minOverlapDutyCycle κάθε στήλης, καθώς και της τιμής inhibitionRadius[49].

- boostStrength (float $\geq 0$): Η παράμετρος αυτή καθορίζει τον βαθμό στον οποίο εφαρμόζεται ο μηχανισμός Ενίσχυσης κατά το πρώτο βήμα εκτέλεσης του αλγορίθμου. Σε περίπτωση που ορίσουμε boostStrength $= 0$, τότε ο μηχανισμός αυτός απενεργοποιείται, καθώς θα ισχύει $b_i = 1$ για κάθε στήλη $i$ της περιοχής, όπου $b_i$ η τιμή ενίσχυσης της στήλης $i$.

- seed (int): Η παράμετρος αυτή αφορά την εκτέλεση των στοχαστικών διαδικασιών του αλγοριθμου.

- wrapAround (bool): Εάν η παράμετρος αυτή λάβει την τιμή «Αληθής» τότε η συμβολοσειρά εισόδου τυλίγεται γύρω από τον εαυτό της, με αποτέλεσμα τα bit τα οποία βρίσκονται στις άκρες της να θεωρούνται «γειτονικά». Αυτό συνεπάγεται την ύπαρξη ορισμένων στηλών συνδεδεμένες με bit εισόδου τα οποία ανήκουν τόσο στην αρχή όσο και στο τέλος της συμβολοσειράς εισόδου.

[49] Γραμμές 929-931 (`https://github.com/numenta/nupic/blob/master/src/nupic/algorithms/spatial_pooler.py`).

Τέλος θα πρέπει να αναφερθούμε στην συνάρτηση *compute* της κλάσης SpatialPooler. Η συνάρτηση αυτή είναι στην ουσία υπεύθυνη για την εκτέλεση του αλγορίθμου, και συνεπώς για την παραγωγή της εξόδου του. Οι παράμετροί της είναι οι εξής:

1. inputVector: Όπως υποδηλώνει το όνομά της η παράμετρος αυτή αναπαριστά την είσοδο στον αλγόριθμο Spatial Pooler, και συνεπώς στην αντίστοιχη περιοχή του συστήματος HTM. Η είσοδος θα πρέπει να αποτελεί μία δυαδική συμβολοσειρά υπό την μορφή λίστας ή πλειάδας, κάθε τιμή της οποίας θα περιέχει και από ένα bit εισόδου. Εναλλακτικά σε περίπτωση που θέλουμε να αποδώσουμε κάποια συγκεκριμένη τοπολογία στην είσοδο, η οργάνωσή της μπορεί να γίνει μέσω NumPy πινάκων.

2. learn: Μία boolean παράμετρος μέσω της οποίας καθορίζουμε εάν το σύστημα HTM βρίσκεται σε λειτουργία εκμάθησης ή όχι. Το τρίτο βήμα του αλγορίθμου, όπως αυτό περιγράφεται στο παραπάνω τμήμα της εργασίας *Ο Αλγόριθμος Spatial Pooler*, θα εκτελεστεί μονάχα εάν η παράμετρος learn λάβει την τιμή «Αληθής».

3. activeArray: Ένας κενός NumPy πίνακας διαστάσεων $N_c \times 1$, όπου $N_c$ το πλήθος των στηλών της περιοχής, ή αλλιώς το μήκος της συμβολοσειράς εξόδου του αλγορίθμου. Μόλις ολοκληρωθεί η εκτέλεση της συνάρτησης compute, ο πίνακας αυτός θα περιέχει στα κελιά του τιμές «0» και «1», οι οποίες υποδηλώνουν τις ανενεργές και ενεργές στήλες αντίστοιχα. Για παράδειγμα εάν το κελί με δείκτη $i$ του πίνακα activeArray περιέχει την τιμή «1», τότε αυτό σημαίνει ότι η αντίστοιχη στήλη με δείκτη $i$ έχει ενεργοποιηθεί. Συνεπώς η παράμετρος αυτή χρησιμοποιείται ως μέσο αποθήκευσης της συμβολοσειράς εξόδου του αλγορίθμου Spatial Pooler.

### Η Κλάση TemporalMemory[50]

Η κλάση TemporalMemory είναι υπεύθυνση για την εκτέλεση του αλγορίθμου Temporal Pooler, ο οποίος περιγράφεται στο τμήμα αυτής της εργασίας με τίτλο *Ο Αλγόριθμος Temporal Pooler*. Παρακάτω παρουσιάζουμε τις παραμέτρους του κατασκευαστή της κλάσης.

- columnDimensions (int ή tuple): Το πλήθος των στηλών μίας περιοχής, ή ισοδύναμα, το μήκος της συμβολοσειράς εισόδου του αλγορίθμου. Τονίζουμε ότι η τιμή αυτής της παραμέτρου θα πρέπει να ταυτίζεται με την τιμή της αντίστοιχης παραμέτρου «columnDimensions» του κατασκευαστή της κλάσης «SpatialPooler».

- cellsPerColumn (int): Το πλήθος των κυττάρων ανά στήλη. Σε συνδυασμό με την παράμετρο «columnDimensions» το μήκος της συμβολοσειράς εξόδου του αλγορίθμου ορίζεται ως columnDimensions · cellsPerColumn.

[50] nupic.algorithms.temporal_memory.TemporalMemory

- activationThreshold (int): Η τιμή αυτή αποτελεί ένα κατώφλι για την ενεργοποίηση των περιφερικών τμημάτων δενδρίτη των κυττάρων. Εάν το πλήθος των ενεργοποιημένων – δηλαδή των ενεργών και συνδεδεμένων– συνάψεων ενός τμήματος δενδρίτη ξεπερνάει αυτό το κατώφλι, τότε το τμήμα αυτό θεωρείται ενεργό.

- initialPermanence (float $\in [0, 1]$): Η τιμή μονιμότητας η οποία αποδίδεται σε μία σύναψη κατά την δημιουργία της[51].

- connectedPermanence (float $\in [0, 1]$): Η τιμή αυτή αποτελεί ένα κατώφλι το οποίο καθορίζει εάν μία σύναψη θεωρείται συνδεδεμένη ή όχι.

- minThreshold (int): Εάν το πλήθος των ενεργών συνάψεων ενός τμήματος δενδρίτη, συνδεδεμένων και μη, είναι μεγαλύτερο ή ίσο της τιμής minThreshold, τότε το τμήμα αυτό θεωρείται συμβατό.

- maxNewSynapseCount (int): Το μέγιστο πλήθος νέων συνάψεων που μπορεί να αποδοθεί σε ένα τμήμα δενδρίτη ανά επανάληψη κατά την διάρκεια της εκπαίδευσης[52]. Θέτοντας την τιμή αυτής της παραμέτρου ίση με το μηδέν επομένως απενεργοποιούμε την εκμάθηση νέων συνάψεων. Αυτό μπορεί να επιταχύνει σε μεγάλο βαθμό την εκτέλεση του αλγορίθμου, μα στην ουσία καταργεί ένα βασικό μέρος της λειτουργίας του αλγορίθμου Temporal Pooler το οποίο αφορά την εκμάθηση εξαρτήσεων μεταξύ των δεδομένων εντός μίας ακολουθίας.

- permanenceIncrement (float $\in [0, 1]$): Τιμή βάσει της οποίας αυξάνεται η τιμή μονιμότητας των ενεργών συνάψεων κατά την διάρκεια της εκπαίδευσης.

- permanenceDecrement (float $\in [0, 1]$): Τιμή βάσει της οποίας μειώνεται η τιμή μονιμότητας των ανενεργών συνάψεων κατά την διάρκεια της εκπαίδευσης.

- predictedSegmentDecrement (float): Παράμετρος βάσει της οποίας «τιμωρούνται» τα τμήματα δενδρίτη των κυττάρων που θεωρούνται ότι έχουν εκτελέσει λανθασμένες προβλέψεις, μειώνοντας την τιμή μονιμότητάς των συνάψεών τους κατά αυτή την τιμή. Σχετικά με τον επιλογή μίας τιμής για αυτήν την παράμετρο η Numenta προτείνει να ορίσουμε την τιμή της ως ελαφρώς υψηλότερη από την τιμή $d \cdot$ permanenceIncrement, όπου $d \in [0, 1]$ η πυκνότητα των ενεργών bit της συμβολοσειράς εισόδου[53].

- seed (int): Η παράμετρος αυτή αφορά την εκτέλεση των στοχαστικών διαδικασιών του αλγοριθμου.

[51] Γραμμή 796 (`https://github.com/numenta/nupic/blob/master/src/nupic/algorithms/temporal_memory.py`)

[52] Κατά την αρχικοποίηση του αλγορίθμου κάθε τμήμα δενδρίτη δημιουργείται χωρίς να του αντιστοιχεί καμία σύναψη: Γραμμή 56 (`https://github.com/numenta/nupic/blob/master/src/nupic/algorithms/connections.py`)

[53] Βλέπε *Η βιβλιοθήκη NuPIC / Η Κλάση SpatialPooler / localAreaDensity*, καθώς και *Η βιβλιοθήκη NuPIC / Η Κλάση SpatialPooler / numActiveColumnsPerInhArea*.

- maxSegmentsPerCell (int): Το μέγιστο πλήθος περιφερικών τμημάτων δενδρίτη που μπορούν να αντιστοιχούν σε κάθε κύτταρο της περιοχής.
- maxSynapsesPerSegment (int): Το μέγιστο πλήθος συνάψεων που μπορούν να αντιστοιχούν σε κάθε περιφερικό τμήμα δενδρίτη των κυττάρων.

Για την περίπτωση της κλάσης TemporalMemory θα πρέπει να εξετάσουμε τρεις διαφορετικές συναρτήσεις. Η πρώτη από αυτές είναι και πάλι η συνάρτηση compute, η οποία όπως και προηγουμένως αφορά ουσιαστικά την εκτέλεση του αλγορίθμου. Οι παράμετροι της συνάρτησης είναι οι εξής:

1. activeColumns: Η παράμετρος αυτή αναπαριστάνει το σύνολο των ενεργών στηλών της περιοχής, δηλαδή την είσοδο στον αλγόριθμο Temporal Pooler και κατ΄ επέκταση στην συνάρτηση compute της κλάσης TemporalMemory. Θα πρέπει να τονίσουμε ότι η παράμετρος θα πρέπει να βρίσκεται υπό την μορφή μίας λίστας (ή ενός μονοδιάστατου NumPy πίνακα), η οποία περιέχει τους δείκτες των ενεργών στηλών. Αυτό σημαίνει ότι η έξοδος της αντίστοιχης συνάρτησης compute της κλάσης SpatialPooler θα πρέπει να μετατραπεί σε αυτην την μορφή πρωτού τροφοδοτηθεί στην συνάρτηση compute της κλάσης TemporalMemory.
2. learn: Μία boolean παράμετρος η οποία καθορίζει εάν τα βήματα του αλγορίθμου που σχετίζονται με την εκπαίδευση του συστήματος, όπως αυτά περιγράφονται στο τμήμα της εργασίας *Ο Αλγόριθμος Temporal Pooler*, εκτελούνται ή όχι.

Αφότου εκτελέσουμε την παραπάνω συνάρτηση στην συνέχεια καλούμε την συνάρτηση *getActiveCells* ούτως ώστε να λάβουμε το σύνολο των δεικτών των ενεργοποιημένων κυττάρων, βάσει των οποίων κατασκευάζεται και η τελική συμβολοσειρά εξόδου όχι μόνο του αλγορίθμου Temporal Pooler αλλά και ολόκληρης της περιοχής.

Η τελευταία συνάρτηση της κλάσης στην οποία θα πρέπει να αναφερθούμε είναι η συνάρτηση *reset*, η οποία καλείται κάθε φορά που το σύστημα πρόκειται να τροφοδοτηθεί με μία νέα ακολουθία δεδομένων. Καλώντας αυτήν την συνάρτηση στην ουσία εξαναγκάζουμε κάθε κύτταρο της περιοχής να εισέλθει σε ανενεργή κατάσταση, είτε αυτό είναι ενεργό είτε σε κατάσταση πρόβλεψης. Κατά αυτόν τον τρόπο θα λέγαμε ότι τα κύτταρα «ξεχνάνε» οποιαδήποτε πληροφορία προέρχεται από την αμέσως προηγούμενη ακολουθία, κάτι το οποίο φυσικά επιθυμούμε εφόσον κάθε ακολουθία θεωρείται ανεξάρτητη από την προηγούμενή της, όπως εξάλλου συμβαίνει και στα πλαίσια των αναδρομικών νευρωνικών δικτύων.

Τέλος θα πρέπει να αναφέρουμε ότι η βιβλιοθήκη NuPIC σαφώς και περιέχει ένα πολύ μεγαλύτερο πλήθος κλάσεων και συναρτήσεων από αυτές που έχουμε περιγράψει έως τώρα. Ωστόσο οι τέσσερεις παραπάνω κλάσεις αρκούν ως προς την κατασκευή αλλά και την εκπαίδευση ενός πλήρως λειτουργικού συστήματος HTM.

# Κεφάλαιο 3

# Μέθοδοι Επίλυσης του Προβλήματος

## 3.1 Το Πρόβλημα προς Επίλυση

Σε αυτήν την εργασία εφαρμόζουμε τα μοντέλα Μηχανικής Μάθησης του αμέσως προηγούμενου κεφαλαίου με σκοπό την ανίχνευση της κακόβουλης διαδικτυακής κυκλοφορίας υπό την μορφή επιθέσεων DDoS. Τέτοιου είδους προβλήματα τις περισσότερες φορές χρήζουν μοντέλων ικανών να διαχειριστούν την όποια χρονική πληροροφορία εμπεριέχεται στα σύνολα δεδομένων. Σαφώς δεν υπάρχει τίποτα που να μας αποτρέπει από την θεώρηση των στοιχείων ενός τέτοιου συνόλου ως μεμονωμένες οντότητες, εκ των οποίων η καθεμία σχετίζεται με κάποια από τις πιθανές κλάσεις του προβλήματος. Παρόλα αυτά η μέθοδος αυτή μεταχείρισης των δεδομένων αγνοεί οποιεσδήποτε τυχόν εξαρτήσεις υπάρχουν μεταξύ των διαφορετικών στοιχείων ενός συνόλου δεδομένων, γεγονός το οποίο ενδέχεται να οδηγήσει σε μειωμένη επίδοση των μοντέλων.

Παραδείγματος χάρη κατά την διάρκεια της διεξαγωγής μίας επίθεσης DDoS είναι βέβαιο πως ο διακομιστής-θύμα θα δεχτεί έναν μεγάλο ογκο πληροφορίας μέσα σε ένα σχετικά μικρό χρονικό διάστημα. Η πληροφορία αυτή είναι επιμερισμένη σε διακριτά πακέτα μεταφοράς δεδομένων, τα οποία εάν δεν είναι πανομοιότυπα μεταξύ τους τότε τουλάχιστον θα μοιράζονται ένα πλήθος κοινών χαρακτηριστικών. Εξετάζοντας κάθε ένα από αυτά τα πακέτα ξεχωριστά, η διαδικασία της ταυτοποίησής τους ως μέρος κακόβουλης δικτυακής κίνησης ενδέχεται να αποτελέσει μία αρκετά επίπονη διαδικασία, καθώς τα πακέτα αυτά είναι εκ των προτέρων κατασκευασμένα κατά τέτοιον τρόπο έτσι ώστε να μην αποκλίνουν σε μεγάλο βαθμό από τα πακέτα εκείνα τα οποία εντάσσονται στην κατηγορία της καλόβουλης κυκλοφορίας. Από την άλλη, θεωρώντας κάθε πακέτο μονάχα ως τμήμα μίας μεγαλύτερης ακολουθίας η οποία εκτείνεται μέσα στον χρόνο, μας παρέχεται η δυνατότητα της συλλογικής εξέτασής τους, ικανή να μας φανερώσει την πραγματική τους σημασία. Για να το εκφράσουμε με πιο απλά λόγια, η οργάνωση των στοιχείων σε χρονικές ακολουθίες

μας επιτρέπει να κάνουμε ένα βήμα πίσω και να δούμε την «μεγάλη εικόνα», βάσει της οποίας γίνεται περισσότερο ξεκάθαρο κατά πόσο ένας διακομιστής βρίσκεται υπό επίθεση ή όχι. Καταλήγουμε συνεπώς πως το χρονικό πλαίσιο μέσα στο οποίο βρίσκεται κάθε στοιχείο του συνόλου δεδομένων αποτελεί μία ιδιαίτερα σημαντική πληροφορία η οποία δεν θα πρέπει σε καμία περίπτωση να αγνοηθεί.

Όπως έχει αναφερθεί και στο δεύτερο κεφάλαιο της εργασίας τα συστήματα HTM καθώς και ορισμένα είδη νευρωνικών δικτύων, π.χ. τα δίκτυα RNN και LSTM, είναι κατάλληλα προσεγμένα για την διαχείριση τέτοιων ακολουθιών δεδομένων. Στόχος μας λοιπόν είναι η εκπαίδευση των παραπάνω μοντέλων κάνοντας χρήση συνόλων δεδομένων, τα στοιχεία των οποίων έχουν οργανωθεί σε επιμέρους χρονικές ακολουθίες ούτως ώστε να οδηγηθούμε σε αλγορίθμους όσο το δυνατόν υψηλότερων επιδόσεων. Ως προς την επίτευξη αυτού του σκοπού αρχικά θα εξετάσουμε μερικές σύγχρονες εργασίες οι οποίες καταπιάνονται με την επίλυση του προβλήματος της ανίχνευσης διαδικτυακών επιθέσεων, εστιάζοντας κατά κύριο λόγο στην χρήση των μοντέλων των νευρωνικών δικτύων. Στην συνέχεια θα παρουσιάσουμε δύο διαφορετικές μεθόδους εφαρμογής των συστημάτων HTM για την αντιμετώπιση του προβλήματος, περιγράφοντας την διαδικασία αυτή στα γενικότερα πλαίσια της επίλυσης προβλημάτων Μηχανικής Μάθησης, συγκεκριμένα του προβλήματος της Ταξινόμησης καθώς και αυτό της Ανίχνευσης Ανωμαλιών.

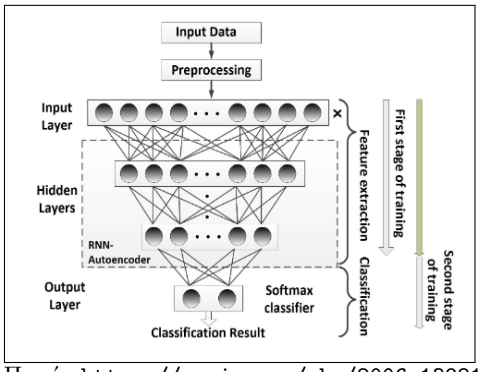
## 3.2 Συναφείς Δουλειές

Όπως και σε αμέτρητες άλλες περιοχές έτσι και στην περιοχή της ασφάλειας δικτύων ένα πλήθος διαφορετικών αλγορίθμων Μηχανικής Μάθησης συνεχίζουν ακατάπαυστα να αξιοποιούνται στην προσπάθεια της εγκαθίδρυσης της ασφαλούς διαδικτυακής επικοινωνίας μεταξύ συσκευών [15, 16, 17]. Ιδιαίτερα τα δύο τελευταία έτη έχει αναδειχθεί σε μεγάλο βαθμό η σημασία της έγκαιρης ανίχνευσης και της αντιμετώπισης των διαδικτυακών επιθέσεων, καθώς λόγω της πανδημίας που διανύουμε η ομαλή λειτουργια των διακομιστών που σχετίζονται με την παροχή υπηρεσιών και προϊόντων μέσω του διαδικτύου είναι πιο σημαντική από ποτέ άλλοτε. Το γεγονός αυτό έχει καταστήσει τέτοιου είδους διακομιστές ως ακόμη μεγαλύτερους στόχους για την διεξαγωγή επιθέσεων [18, 19], κάτι το οποίο έχει οδηγήσει στην δημοσίευση μίας πληθώρας σχετικών με το πρόβλημα εργασιών σε ένα μικρό χρονικό διάστημα.

Συγκεκριμένα η χρήση των μοντέλων των Αυτοκωδικοποιητών φαίνεται να κατέχει μία δημοφιλή θέση μεταξύ των αλγορίθμων επίλυσης του προβλήματος. Ένα παράδειγμα αποτελεί η εργασία [20] η οποία προτείνει το *DDoSNet*, ένα δίκτυο το οποίο συνδυάζει την έννοια του Αυτοκωδικοποιητή και των RNN. Στην ουσία πρόκειται για έναν Αυτοκωδικοποιητή αποτελούμενο από οκτώ[1] κρυφά επίπεδα, καθένα από τα οποία ορίζεται ως μία ξεχωριστή RNN μονάδα. Μέσω αυτής της αρχιτεκτονικής το δίκτυο είναι ικανό να εκμεταλλευτεί την χρονική πληροφορία των δεδομένων.

[1] Τα τέσσερα πρώτα επίπεδα αποτελούν τον κωδικοποιητή του δικτύου, ενώ τα τέσσερα τελευταία αποτελούν τον αποκωδικοποιητή αντίστοιχα.

Σχήμα 3.1: Το Μοντέλο *DDoSNet*.




Πηγή: `https://arxiv.org/abs/2006.13981`

Η εκπαίδευση του μοντέλου *DDoSNet* διασπάται σε δύο στάδια: Το πρώτο από αυτά αφορά την μη-επιβλεπόμενη εκπαίδευση του δικτύου κατά την οποία τα βάρη του προσαρμόζονται με σκοπό την χαμηλού σφάλματος ανακατασκευή της εισόδου. Στην συνέχεια ακολουθεί η φάση της επιβλεπόμενης μάθησης, κατά το ξεκίνημα της οποίας ο αποκωδικοποιητής του δικτύου αντικαθίσταται από ένα απλό επίπεδο εξόδου δύο νευρώνων, κάθε ένας από τους οποίους αντιστοιχεί και σε μία διαφορετική κλάση, δηλαδή στην καλόβουλη και στην κακόβουλη διαδικτυακή κίνηση. Το επίπεδο αυτό εκπαιδεύεται ξεχωριστά κάνοντας χρήση των ετικετών των δεδομένων με σκοπό τον υπολογισμό των συναπτικών βαρών που οδηγούν στην επιτυχή ταξινόμηση των στοιχείων του συνόλου εκπαίδευσης. Ως συνάρτηση ενεργοποίησης του επιπέδου χρησιμοποιείται η συνάρτηση Softmax, παρέχοντας έτσι στο δίκτυο την δυνατότητα να αποδίδει σε οποιοδήποτε στοιχείο εισόδου δύο τιμές, οι οποίες εκφράζουν τις πιθανότητες του στοιχείου αυτού να ανήκει σε κάθε μία από τι δύο κλάσεις. Αξίζει επίσης να τονίσουμε ότι τα δεδομένα τα οποία σχετίζονται με την κακόβουλη κυκλοφορία δεν προέρχονται μονάχα από μία, αλλά από περισσότερες από δέκα διαφορετικές επιθέσεις τύπου DDoS, μερικές από τις οποίες εμπεριέχονται είτε μονάχα μέσα στο σύνολο εκπαίδευσης είτε στο σύνολο αξιολόγησης. Παρόλα αυτά το μοντέλο επιτυγχάνει πάνω στο πρόβλημα ακρίβεια της τάξης του 99%, υπερνικώντας διάφορα άλλα μοντέλα Μηχανικής Μάθησης, όπως για παράδειγμα τα μοντέλα «Support Vector Machine» και τα «Δέντρα Αποφάσεων» (Decision Trees).

Στα πλαίσια της αντιμετώπισης του ίδιου προβλήματος εξίσου αξιοσημείωτα είναι τα αποτελέσματα της εργασίας [21], η οποία αν και αξιοποιεί ένα μοντέλο Αυτοκωδικοποιητή απλών πλήρως συνδεδεμένων κρυφών επιπέδων παράλληλα καταβάλλεται μεγαλύτερη προσπάθεια ως προς την κατασκευή συνόλων δεδομένων, τα χαρακτηριστικά των οποίων ενσωματώνουν κατά βάση την χρονική αλληλεξάρτηση των πακέτων. Τα αποτελέσματα των πειραμάτων που παρουσιάζονται εντός της εργασίας αποδεικνύουν ότι ακόμη και για ένα μοντέλο το οποίο είναι ανίκανο να διαχειριστεί χρονικές ακολουθίες, η κατόρθωση υψηλών επιδόσεων πάνω σε τέτοιου είδους προβλήματα παραμένει εφικτή αρκεί η πληροφορία αυτή να εκφράζεται επαρκώς μέσω των δεδομένων. Θα πρέπει επιπλέον να τονίσουμε ότι το συγκεκριμένο μοντέλο δεν κάνει χρήση των ετικετών των δεδομένων ως προς την εκαίδευσή του, καθώς αυτή πραγματοποιείται βάσει δεδομένων τα οποία αντιπροσωπεύουν την καλόβουλη διαδικτυακή κίνηση και μόνο, αποβλέποντας στην ελαχιστοποίηση του σφάλματος ανακατασκευής. Ωστόσο οι ετικέτες χρησιμοποιούνται εκ των υστέρων με σκοπό τον υπολογισμό μίας τιμής κατωφλιού βάσει της οποίας

το μοντέλο κρίνει εάν ένα στοιχείο εισόδου αποτελεί «ανωμαλία» ή όχι. Εάν η τιμή του σφάλματος ανακατασκευής ενός στοιχείου ξεπεράσει την τιμή κατωφλιού, τότε το μοντέλο το ταξινομεί στην κλάση της κακόβουλης κίνησης. Μέσω της χρήσης ενός κατωφλιού το πρόβλημα της Ανίχνευσης Ανωμαλιών μετατρέπεται συνεπώς σε ένα κλασικό πρόβλημα Δυαδικής Ταξινόμησης.

Παρόλα αυτά υπάρχουν και άλλες προσεγγίσεις οι οποίες δεν βασίζονται στην χρονική πληροφορία των δεδομένων, όπως για παράδειγμα η εργασία [22] η οποία αφορά την εφαρμογη ενός μοντέλου *ResNet* το οποίο διαχειρίζεται οπτικές αναπαραστάσεις της διαδικτυακής κίνησης, κατασκευασμένες βάσει ροών πακέτων. Από ό,τι παρατηρούμε λοιπόν υπάρχει ένα ευρύ σύνολο διαφορετικών προσεγγίσεων, γεγονός το οποίο αποτελεί ένδειξη της κρισιμότητας του προβλήματος. Όσο αφορά την δική μας συνεισφορά, τα μοντέλα που χρησιμοποιούμε αξιολογούνται τόσο στα πλαίσια της Επιβλεπόμενης όσο και στα πλαίσια της Μη-Επιβλεπόμενης Μάθησης, υπό την μορφή των προβλημάτων Ταξινόμησης και Ανίχνευσης Ανωμαλιών αντίστοιχα. Και για τις δύο περιπτώσεις επίλυσης των προβλημάτων θεωρείται ωστόσο ότι η διαδικασία εφαρμογής των μοντέλων των νευρωνικών δικτύων δεν χρήζει περαιτέρω διερεύνησης –πέρα των όσων έχουμε αναφέρει στο δεύτερο κεφάλαιο της εργασίας– παρά μόνο αναφοράς των αρχιτεκτονικών που αξιοποιούνται σε κάθε περίπτωση, οι οποίες περιγράφονται στο τέταρτο κεφάλαιο. Για την περίπτωση των συστημάτων HTM παρόλα αυτά, όντας μοντέλα λιγότερο διαδεδομένα, αφιερώνουμε παρακάτω ένα ξεχωριστό τμήμα έτσι ώστε να εμβαθύνουμε στην μεθοδολογία βάσει της οποίας κάνουμε χρήση αυτού του είδους μοντέλων με σκοπό την επίλυση των παραπάνω προβλημάτων.

## 3.3 Εφαρμογή των Συστημάτων HTM σε προβλήματα Μηχανικής Μάθησης

Όπως έχουμε αναφέρει δυό είναι τα προβλήματα για την επίλυση των οποίων επιχειρούμε να εφαρμόσουμε τα συστήματα HTM. Το πρώτο από αυτά αποτελεί το πρόβλημα της Ταξινόμησης, ενώ το δεύτερο ορίζεται ως το πρόβλημα της Ανίχνευσης Ανωμαλιών. Φυσικά, ο τελικός σκοπός παραμένει ο ίδιος και για τις δύο περιπτώσεις, και αυτός είναι η κατασκευή ενός μοντέλου ικανού να διαχωρίζει την καλόβουλη από την κακόβουλη κίνηση. Παρόλα αυτά η περίπτωση της ανίχνευσης ανωμαλιών θα λέγαμε ότι αποτελεί μία περισσότερο έμεση προσέγγιση επίλυσης του προβλήματος, η οποία λειτουργεί μονάχα επειδή το αντιστοιχο πρόβλημα ταξινόμησης είναι δυαδικό, δηλαδή αφορά αποκλειστικά και μόνο δύο κλάσεις –την καλόβουλη και την κακόβουλη διαδικτυακή κίνηση. Συνεπώς μπορούμε να κάνουμε την παραδοχή ότι τα ανώμαλα στοιχεία που ανιχνεύονται από το σύστημα HTM θα αποτελούν μέρος της κακόβουλης κίνησης, επιλύοντας έτσι στην ουσία το πρόβλημα της Ταξινόμησης. Παρακάτω εξετάζουμε ξεχωριστά τις μεθόδους βάσει των οποίων εφαρμόζουμε τα συστήματα HTM για την επίλυση κάθε ενός από τα δύο παραπάνω προβλήματα.

### 3.3.1 Το Πρόβλημα της Ταξινόμησης

Η μέθοδος που εφαρμόζουμε ως προς την επίλυση του προβλήματος της Ταξινόμησης παραπέμπει στον «Αλγόριθμο Φλοιού Μάθησης» (Cortical Learning Algorithm - CLA) της Numenta. Στην πράξη η ταξινόμηση κάθε στοιχείου πραγματοποιείται ακολουθώντας πιστά τα βήματα του εν λόγω αλγορίθμου, με την μόνη διαφορά ότι ενώ ο αλγόριθμος CLA απαιτεί την ένταξη του χαρακτηριστικού προς πρόβλεψη –στην περίπτωσή μας των ετικετών– στις κωδικοποιημένες συμβολοσειρές SDR των στοιχείων εισόδου[2], η μέθοδος η οποία προτείνεται σε αυτήν την εργασία δεν περιορίζεται από κάτι τέτοιο. Θα πρέπει βέβαια να τονίσουμε ότι με την εφαρμογή της μεθόδου μας χάνεται η ικανότητα των μοντέλων να εκτελούν μελλοντικές προβλέψεις, κάτι το οποίο ναι μεν είναι εφικτό μέσω του αλγορίθμου CLA αλλά στην προκειμένη περίπτωση δεν μας απασχολεί.

Η μέθοδος βάσει της οποίας ένα σύστημα HTM προβλέπει την κλάση ενός στοιχείου εισόδου είναι στην πραγματικότητα αρκετά απλή και κατανοητή. Υπενθυμίζουμε ότι η έξοδος του μοντέλου αποτελεί μονάχα μία δυαδική συμβολοσειρά, η οποία υποδεικνύει τα ενεργά κύτταρα της τελευταίας περιοχής του συστήματος. Βάσει αυτής της συμβολοσειράς εξόδου ο αλγόριθμος ταξινομεί το στοιχείο στην είσοδο εξετάζοντας το σύνολο των ενεργοποιημένων κυττάρων, καθώς και την γενικότερη συχνότητα ενεργοποίησής τους η οποία έχει θα πρέπει να έχει καταμετρηθεί εκ των προτέρων κατά την διάρκεια της εκπαίδευσης.

Πιο συγκεκριμένα ο αλγόριθμος διατηρεί έναν πίνακα $\boldsymbol{C}$ διαστάσεων $c \times n$, όπου $c$ το πλήθος των διαφορετικών κλάσεων του εκάστοτε προβλήματος, ενώ $n$ το μήκος της συμβολοσειράς εξόδου, δηλαδή το συνολικό πλήθος των κυττάρων της τελευταίας περιοχής του συστήματος. Κάθε γραμμή $\boldsymbol{C}_{i,:}$ του πίνακα $\boldsymbol{C}$ αντιστοιχεί και σε μία διαφορετική κλάση $i$, ενώ κάθε τιμή $\boldsymbol{C}_{i,j}$ εκφράζει το πλήθος των επαναλήψεων κατά την διάρκεια της εκπαίδευσης του συστήματος για τις οποίες το κύτταρο $j$ ήταν ενεργό, δεδομένου ότι το εκάστοτε στοιχείο εισόδου που οδήγησε στην ενεργοποίησή του ανήκε στην κλάση $i$. Συνεπώς μέσω αυτού του πίνακα ο αλγόριθμος επιχειρεί να «μάθει» ποιες κλάσεις οδηγούν στην ενεργοποίηση ποιανών κυττάρων, και με τί συχνότητα.

Αφότου ολοκληρωθεί η εκπαίδευση του συστήματος βάσει του εκάστοτε συνόλου δεδομένων, ο αλγόριθμος διαιρεί τις τιμές κάθε στήλης $\boldsymbol{C}_{:,j}$ του πίνακα με το άθροισμα τους –το μέγεθος αυτό εκφράζει την συνολική συχνότητα ενεργοποίησης του κυττάρου $j$– έτσι ώστε κάθε στήλη να μετατραπεί σε μία κατανομή πιθανοτήτων[3]. Οι τιμές κάθε στήλης $\boldsymbol{C}_{:,j}$ του πίνακα επομένως πλέον εκφράζουν την πιθανότητα κάθε κλάσης $i$ δεδομένου ότι το κύτταρο $j$ είναι ενεργό. Μετά την πραγματοποίηση των παραπάνω το μοντέλο θεωρείται πλήρως εκπαιδευμένο και μπορεί να εκτελεί προβλέψεις σχετικά με την κλάση ενός νέου στοιχείου στην

[2]Βάσει μίας τέτοιας μεθόδου είναι προφανές πως θα ήταν απαραίτητο να δεσμεύσουμε μία επιπλέον τιμή για το χαρακτηριστικό προς ταξινόμηση, με σκοπό την κωδικοποίηση των αγνώστων στοιχείου εισόδου χωρίς να τα εντάξουμε εκ των προτέρων σε κάποια κλάση.

[3]Έαν στα πλαίσια της εκπαίδευσης ένα κύτταρο $j$ δεν έχει ενεργοποιηθεί καμία φορά, με αποτέλεσμα να ισχύει $\boldsymbol{C}_{i,j} = 0$, τότε θέτουμε τις αντίστοιχες τιμές της στήλης $\boldsymbol{C}_{:,j}$ ως μία πολύ μικρή τιμή, π.χ. 0.0001.

είσοδο ακολουθώντας τα τρία παρακάτω βήματα[4]:

1. Δεδομένου ενός στοιχείου στην είσοδο το σύστημα HTM παράγει την συμβολοσειρά εξόδου η οποία υποδεικνύει τα ενεργά κύτταρα της τελευταίας περιοχής του συστήματος.

2. Έστω $A$ το σύνολο το οποίο περιέχει τους δείκτες κάθε ενεργού κυττάρου. Για κάθε κλάση $i$ του προβλήματος υπολογίζουμε την εξής τιμή:

$$p_i = \prod_{j \in A} \boldsymbol{C}_{i,j}$$

3. Τέλος ο αλγόριθμος προβλέπει την κλάση του στοιχείου εισόδου ως την κλάση $k$, στην οποία αντιστοιχεί η υψηλότερη μεταξύ των τιμών $p_i$, δηλαδή:

$$k = \arg\max_i (p_i)$$

Θα πρέπει να τονίσουμε ότι αν και τα μεγέθη των τιμών $p_i$ υποδεικνύουν σε ποιά από τις κλάσεις είναι πιο πιθανό να ανήκει το στοιχείο εισόδου, οι τιμές αυτές καθαυτές δεν αποτελούν πιθανότητες. Παρόλα αυτά σε κάθε περίπτωση μπορούμε να διαιρέσουμε κάθε τιμή $p_i$ με το συνολικό άθροισμα τους, έτσι ώστε να κατασκευάσουμε μία τέτοια πιθανοτική κατανομή. Μέσω της παραπάνω διαδικασίας επομένως ο αλγόριθμος είναι ικανός να ταξινομήσει οποιοδήποτε νέο στοιχείο εισόδου. Σε αυτό το σημείο να σημειώσουμε επίσης ότι μετά το πέρας της εκπαίδευσης ο πίνακας $\boldsymbol{C}$ δεν θα πρέπει να τροποποιηθεί περαιτέρω, δηλαδή η διαδικασία καταμέτρησης της συχνότητας ενεργοποίησης των κυττάρων θεωρείται ολοκληρωμένη.

### 3.3.2 Το Πρόβλημα της Ανίχνευσης Ανωμαλιών

Για την επίλυση του δεύτερου προβλήματος το σύστημα HTM θα πρέπει να είναι ικανό να βαθμολογεί κάθε στοιχείο εισόδου αποδίδοντάς του μία τιμή $a \in [0, 1]$, η οποία εκφράζει την πιθανότητα του εκάστοτε στοιχείου να αποτελεί ανωμαλία, πάντοτε συγκριτικά με τα δεδομένα βάσει των οποίων ο αλγόριθμος έχει εκπαιδευτεί. Τροφοδοτώντας επομένως στον αλγόριθμο στοιχεία τα οποία σχετίζονται αποκλειστικά με την καλόβουλη δικτυακή κίνηση, μπορούμε μετέπειτα να θεωρήσουμε ότι κάθε «ανώμαλο» στοιχείο που ανιχνεύεται από το σύστημα αποτελεί σημάδι κακόβουλης κίνησης. Παρατηρούμε λοιπόν ότι το πρόβλημα αυτό ανήκει στην κατηγορία των προβλημάτων Μη-Επιβλεπόμενης Μάθησης, εφόσον η ύπαρξη των αντίστοιχων ετικετών των στοιχείων του συνόλου εκπαίδευσης δεν είναι απαραίτητη[5].

[4] Μία ακόμη διαφορά μεταξύ της μεθόδου μας και του αρχικού αλγορίθμου CLA αφορά το δεύτερο από τα παρακάτω βήματα, για το οποίο ο αλγόριθμος CLA υπολογίζει τις τιμές $p_i$ ως το άθροισμα αντί το γινόμενο των τιμών $\boldsymbol{C}_{i,j}$.

[5] Να σημειωθεί ότι παρόλο που οι ετικέτες δεν είναι απαραίτητες, θα πρέπει να γνωρίζουμε εκ των προτέρων ότι τα δεδομένα εκπαίδευσης αποτελούν μέρος καλόβουλης κίνησης.

Πως όμως ένα σύστημα HTM υπολογίζει έναν «βαθμό ανωμαλίας» για ένα στοιχείο στην είσοδο; Στην περίπτωση αντιμετώπισης αυτού του προβλήματος η διαδικασία που ακολουθούμε ταυτίζεται απόλυτα με την μέθοδο που προτείνεται εντός του βιβλίου *Anti-fragile ICT Systems* [23], η οποία βασίζεται στον μηχανισμό Έκρηξης των στηλών. Όπως έχουμε αναφέρει στο δεύτερο κεφάλαιο της εργασίας, μία απο τις τρεις πιθανές καταστάσεις των κυττάρων αποτελεί και η «κατάσταση πρόβλεψης», μέσω της οποίας το σύστημα εκφράζει τις προβλέψεις που εκτελεί αναφορικά με το αμέσως επόμενο στοιχείο εισόδου. Όταν ένα ή περισσότερα κύτταρα μίας στήλης $i$ βρίσκονται σε κατάσταση πρόβλεψης κατά την $t$-οστή επανάληψη του αλγορίθμου, το γεγονός αυτό μπορεί να ερμηνευτεί ως η πρόβλεψη του συστήματος πως το αμέσως επόμενο στοιχείο εισόδου ενδέχεται να οδηγήσει στην ενεργοποίηση της στήλης $i$. Εάν επομένως κατά την επανάληψη $t + 1$ πράγματι παρατηρηθεί η ενεργοποίηση της εν λόγω στήλης, τότε η πρόβλεψη του συτστήματος θεωρείται επιτυχής με τον αλγόριθμο Temporal Pooler να προβαίνει στην ενεργοποίηση των κυττάρων της στήλης τα οποία βρίσκονται σε κατάσταση πρόβλεψης. Εάν από την άλλη η στήλη δεν ενεργοποιηθεί, τότε αυτό σημαίνει πως η πρόβλεψη αποδείχθηκε λανθασμένη, με αποτέλεσμα την μείωση των τιμών μονιμότητάς των συνάψεών που συνέβαλλαν στην εκτέλεσή της ως ένα είδος «τιμωρίας».

Στην περίπτωση όμως που ενεργοποιείται μία στήλη η οποία δεν περιέχει κανένα κύτταρο σε κατάσταση πρόβλεψης, τότε εκτελείται ο μηχανισμός Έκρηξης μέσω του οποίου ενεργοποιείται κάθε κύτταρό της. Η πυροδότηση αυτού του μηχανισμού υποδηλώνει ότι το μοντέλο δέχεται μη αναμενόμενη πληροφορία, την οποία ενδεχομένως να μην έχει συναντήσει ποτέ ξανά στο παρελθόν. Κατά την πρώιμη φάση της εκπαίδευσης του συστήματος η υψηλή συχνότητα εκτέλεσης του μηχανισμού Έκρηξης θεωρείται φυσιολογική καθώς το μοντέλο δεν παύει να ανακαλύπτει νέα μοτίβα. Μέσω του μηχανισμού Έκρηξης το σύστημα οδηγείται στην ενεργοποίηση περισσότερων κυττάρων, κάτι το οποίο αυξάνει την πιθανότητα των κυττάρων της περιοχής να εισέλθουν σε κατάσταση πρόβλεψης, το οποίο με την σειρά του οδηγεί σε ταχύτερη εκπαίδευση του συστήματος, καθώς όσα περισσότερα κύτταρα βρίσκονται σε κατάσταση πρόβλεψης, τόσες περισσότερες διαδικασίες εκπαίδευσης πραγματοποιούνται κατά την εκτέλεση του αλγορίθμου Temporal Pooler.

Παρόλα αυτά, υπό τις συνθήκες κατά τις οποίες ένα πλήρως εκπαιδευμένο μοντέλο δέχεται δεδομένα συνεπή με τα δεδομένα εκείνα βάσει των οποίων έχει εκπαιδευτεί, ο μηχανισμός Έκρηξης δεν θα πρέπει να αποτελεί ένα τόσο συνηθισμένο φαινόμενο. Για αυτόν τον λόγο ο μηχανισμός αυτός θεωρείται ως μία ένδειξη ότι το σύστημα πιθανώς να έχει δεχτεί ένα στοιχείο εισόδου το οποίο παρουσιάζει ιδιομορφίες. Συγκεκριμένα όσο περισσότερες στήλες πυροδοτούν τον μηχανισμό Έκρηξης, τόσο πιο πιθανό είναι το εκάστοτε στοιχείο εισόδου να αποτελεί ανωμαλία. Βάσει αυτής της παρατήρησης ο βαθμός ανωμαλίας $a$ ενός στοιχείου εισόδου υπολογίζεται ως εξής:

$$a = \frac{\text{Πλήθος των ενεργών στηλών που πυροδοτούν τον μηχανισμό Έκρηξης}}{\text{Συνολικό πλήθος των ενεργών στηλών}}$$

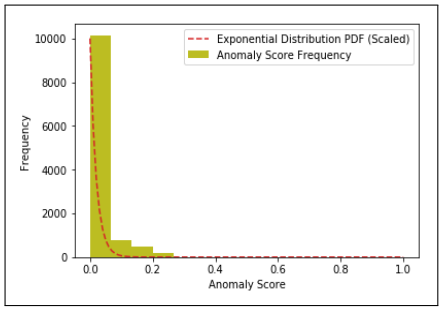
Εφόσον το πλήθος των στηλών που πυροδοτούν τον μηχανισμό Έκρηξης θα είναι

πάντοτε μικρότερο ή ίσο του συνολικού πλήθους των ενεργών στηλών, ο παραπάνω τύπος εγγυάται πως κάθε τιμή $a$ θα ανήκει στο διάστημα τιμών $[0, 1]$, δηλαδή στην ουσία θα αποτελεί πιθανότητα. Εάν για ένα στοιχείο εισόδου $x_j$ ισχύει ότι $a_j = 0$ τότε αυτό σημαίνει ότι το μοντέλο προέβλεψε με επιτυχία την ενεργοποίηση κάθε στήλης. Από την άλλη εάν η τιμή $a_j$ ισούται με την μονάδα τότε αυτό συνεπάγεται πως οι προβλέψεις του μοντέλου ήταν όλες τους λανθασμένες, κάτι το οποίο πιθανότατα οφείλεται στην ιδιομορφία του εκάστοτε στοιχείου εισόδου.

### 3.3.3 Από την Ανίχνευση Ανωμαλιών στην Δυαδική Ταξινόμηση

Έχοντας περιγράψει την διαδικασία υπολογισμού του βαθμού ανωμαλίας στην συνέχεια θα πρέπει να ορίσουμε μία τιμή κατωφλιού βάσει της οποίας θα αποφασίζεται κατά πόσο ένα στοιχείο αντιστοιχεί σε καλόβουλη ή σε κακόβουλη κίνηση. Μέσω αυτής της διαδικασίας το πρόβλημα Ανίχνευσης Ανωμαλιών επομένως μετατρέπεται σε ένα πρόβλημα Δυαδικής Ταξινόμησης. Σε αντίθεση με την εργασία [21] βάσει της οποίας ο υπολογισμός του κατωφλιού χρήζει των ετικετών των δεδομένων, η μέθοδος η οποία προτείνουμε αγνοεί ολότελα την ύπαρξή τους. Να σημειωθεί επίσης ότι η διαδικασία υπολογισμού της τιμής κατωφλιού που περιγράφουμε παρακάτω χρησιμοποιείται στα πλαίσια εφαρμογής τόσο των συστημάτων HTM όσο και των μοντέλων των νευρωνικών δικτύων.

Η μέθοδος υπολογισμού της τιμής κατωφλιού βασίζεται στην υπόθεση ότι οι βαθμοί ανωμαλίας που εξάγει ένα πλήρως εκπαιδευμένο μοντέλο βάσει των στοιχείων εκπαίδευσής του θα ακολουθούν την εκθετική κατανομή με συνάρτηση πυκνότητας πιθανότητας $p(x; \lambda) = \lambda e^{-\lambda x}$ αγνώστης παραμέτρου $\lambda > 0$. Δεδομένου ότι το σύνολο εκπαίδευσης είναι αντιπροσωπευτικό ολόκληρου του πληθυσμού των δεδομένων της κλάσης, είναι λογικό να υποθέσουμε ότι οι βαθμοί ανωμαλίας των στοιχείων θα ακολουθούν μία εκθετική κατανομή. Στο Σχήμα 3.2 απεικονίζουμε το ιστόγραμμα των βαθμών ανωμαλίας που εξάγει ένα σύστημα HTM βάσει 11,547 καλόβουλων στοιχείων ροών πακέτων, αφότου έχει προηγουμένως εκπαιδευτεί βάσει αυτών. Οι τιμές τους φαίνεται πράγματι να ακολουθούν μία εκθετική κατανομή, ανγώστης ωστόσο παραμέτρου $\lambda$.




Σχήμα 3.2: Ιστόγραμμα του βαθμού ανωμαλίας που εξάγει ένα σύστημα HTM για 11,547 στοιχεία ροών πακέτων κάνοντας χρήση κάδων μεγέθους 0.1.

Έχοντας κάνει την παραπάνω υπόθεση υπολογίζουμε την τιμή κατωφλιού βάσει του κριτηρίου του Tukey [24] ο οποίος ισχυρίζεται ότι οι ανώμαλες τιμές μίας κατανομής βρίσκονται πέρα του σημείου που ορίζεται ως το άθροισμα Q3 + IQR, όπου Q3 το τρίτο τεταρτημόριο της κατανομής, και IQR = Q3 − Q1 το διατεταρ-

τημοριακό διάστημα. Συγκεκριμένα για την περίπτωση της εκθετικής κατανομής το σημείο αυτό, έστω $\theta_a$, υπολογίζεται βάσει του εξής τύπου:

$$\theta_a = \frac{\ln(4) + 1.5 \cdot \ln(3)}{\lambda}$$

Συνεπώς το μόνο που απομένει είναι να υπολογίσουμε την παράμετρο $\lambda$ της εκθετικής κατανομής των βαθμών ανωμαλίας που εξάγει το μοντέλο. Για να λύσουμε αυτό το πρόβλημα χρησιμοποιούμε την μέθοδο της «Εκτίμησης Μέγιστης Πιθανοφάνειας» (Maximum Likelihood Estimation), ορίζοντας ως τελική τιμή $\lambda$ την τιμη $\hat{\lambda}$ η οποία μεγιστοποιεί την πιθανοφάνεια των βαθμών ανωμαλίας, υπό την υπόθεση ότι οι τιμές τους ακολουθούν την εκθετική κατανομή. Πιο συγκεκριμένα, δεδομένου ενός συνόλου βαθμών ανωμαλίας $A = \{a_1, a_2, ..., a_n \mid a_i \geq 0\}$, με τις τιμές $a_i$ του συνόλου να προέρχονται από την ίδια εκθετική κατανομή αγνώστης παραμέτρου $\lambda$, η πιθανοφάνεια $L_n(\lambda; A)$ των τιμών του συνόλου $A$ υπολογίζεται ως εξής:

$$L_n(\lambda; A) = p(A; \lambda) = \prod_{i=1}^{n} p(a_i; \lambda) = \prod_{i=1}^{n} \lambda e^{-\lambda \cdot a_i} = \lambda^n \cdot \exp\left(-\lambda \sum_{i=1}^{n} a_i\right)$$

Ωστόσο συνηθίζεται να χρησιμοποιούμε τον λογάριθμο της πιθανοφάνειας $\ln(L_n)$ αντί της ίδιας της πιθανοφάνειας, καθώς αυτή η πρακτική οδηγεί σε απλούστερους υπολογισμούς. Το αποτέλεσμα βέβαια παραμένει το ίδιο καθώς η λογαριθμική συνάρτηση είναι γνησίως αύξουσα:

$$\begin{aligned}
\ln\left[L_n(\lambda; A)\right] &= \ln\left[\lambda^n \cdot \exp\left(-\lambda \sum_{i=1}^{n} a_i\right)\right] \\
&= \ln\left(\lambda^n\right) + \ln\left[\exp\left(-\lambda \sum_{i=1}^{n} a_i\right)\right] \\
&= n \cdot \ln(\lambda) - \lambda \sum_{i=1}^{n} a_i
\end{aligned}$$

Η τελική τιμή $\hat{\lambda}$ επομένως υπολογίζεται ως:

$$\hat{\lambda} = \arg\max_{\lambda > 0} \left[n \cdot \ln(\lambda) - \lambda \sum_{i=1}^{n} a_i\right]$$

Πλέον μπορούμε να ορίσουμε το κατώφλι $\theta_a$, βάσει του οποίου κάθε στοιχείο θα ταξινομείται σε μία από τις δύο κλάσεις:

$$\theta_a = \frac{\ln(4) + 1.5 \cdot \ln(3)}{\hat{\lambda}}$$

Η παραπάνω διαδικασία μας παρέχει δύο πλεονεκτήματα: Το πρώτο και κυριότερο είναι ότι δεν χρειαζόμαστε τις ετικέτες των δεδομένων, συνεπώς η μέθοδος επίλυσης του προβλήματος παραμένει εξ ολοκλήρου στα πλαίσια της Μη-Επιβλεπόμενης Μάθησης. Το δεύτερο πλεονέκτημα αποτελεί το γεγονός ότι η

τιμή του κατωφλιού υπολογίζεται βάσει των δεδομένων του συνόλου εκπαίδευσης, δηλαδή η μέθοδος δεν μας εξαναγκάζει να αναζητήσουμε περαιτέρω δεδομένα. Θα πρέπει επίσης να τονίσουμε ότι κατά τον υπολογισμό της τιμής της παραμέτρου $\lambda$ προτιμάμε να αφαιρούμε υπερβολικά υψηλούς βαθμούς ανωμαλίας ($\geq 0.9$) –εάν αυτοί υπάρχουν– καθώς η ύπαρξή τους ενδέχεται εσφαλμένα να οδηγήσει σε υψηλότερες τιμές του κατωφλιού.

Τέλος θα πρέπει να αναφερθούμε σε μία παρατήρηση η οποία αφορά αποκλειστικά και μόνο τα συστήματα HTM και τον μηχανισμό Έκρηξης. Όπως είδαμε ο μηχανισμός αυτός κατέχει έναν κεντρικο ρόλο στον υπολογισμό των βαθμών ανωμαλίας των στοιχείων εισόδου, καθώς όσες περισσότερες φορές ένα στοιχείο πυροδοτήσει τον μηχανισμό, τόσο πιο πολύ αυξάνεται η πιθανότητα του στοιχείου αυτού να αποτελεί ανωμαλία. Ωστόσο στο δεύτερο κεφάλαιο της εργασίας έχουμε αναφέρει πως κατά την τροφοδότηση του συστήματος HTM με το πρώτο στοιχείο μίας νέας ακολουθίας, ο μηχανισμός Έκρηξης είναι βέβαιο πως θα πυροδοτηθεί αφού κανένα κύτταρο της περιοχής δεν πρόκειται να βρίσκεται σε κατάσταση πρόβλεψης, η οποία θέτει κάθε κύτταρο της περιοχής ως ανενεργό. Αυτό συνεπάγεται ότι κάθε στοιχείο εισόδου το οποίο αποτελεί το πρώτο στοιχείο της ακολουθίας στην οποία εμπεριέχεται είναι καταδικασμένο να λαμβάνει πάντοτε βαθμό ανωμαλίας ίσο με την μονάδα, κάτι το οποίο προφανώς δεν συμβαδίζει με την πραγματικότητα. Για αυτόν τον λόγο, στα πλαίσια της ανίχνευσης ανωμαλιών θα πρέπει να αποφεύγουμε την οργάνωση των δεδομένων σε χρονικές ακολουθίες, το οποίο πολύ απλά στην πράξη σημαίνει ότι η συνάρτηση *reset* της κλάσης *TemporalMemory* δεν θα πρέπει να καλείται κατά το ξεκίνημα κάθε νέας ακολουθίας.

# Κεφάλαιο 4

# Πειράματα και Αποτελέσματα

Στο κεφάλαιο αυτό θα ασχοληθούμε με τα πειράματα που εκτελέστηκαν στα πλαίσια αυτής της εργασίας, καθώς και με οποιαδήποτε άλλη διαδικασία σχετίζεται με αυτά. Συγκεκριμένα θα ξεκινήσουμε παρέχοντας πληροφορίες αναφορικά με τα σύνολα δεδομένων που χρησιμοποιήθηκαν στα πλαίσια των πειραμάτων, εστιάζοντας παράλληλα στο σύνολο των χαρακτηριστικών που χρησιμοποιούμε σε κάθε περίπτωση. Εν συνεχεία παρουσιάζουμε κάθε μοντέλο που χρησιμοποιούμε ως προς την εκτέλεση των πειραμάτων. Στο μέρος αυτό εμπεριέχονται πληροφορίες οι οποίες αφορούν τόσο την αρχιτεκτονική όσο και τις τιμές των παραμέτρων κάθε μοντέλου. Αμέσως μετά ακολουθεί η περιγραφή της φάσης προετοιμασίας των πειραμάτων, η οποία εμπεριέχει την κατασκευή και προεπεξεργασία των συνόλων εκπαίδευσης και αξιολόγησης, ενώ στην συνέχεια ακολουθεί μία περιγραφή των ίδιων των πειραμάτων. Τέλος το κεφάλαιο αυτό κλείνει με την παρουσίαση και τον σχολιασμό των αποτελεσμάτων που εξήχθησαν μέσω της εκτέλεσης των εν λόγω πειραμάτων.

## 4.1 Τα Σύνολα Δεδομένων

Τα πειράματα που εκτελούμε αφορούν δύο διαφορετικά είδη δεδομένων, κάθε ένα από τα οποία αντιστοιχεί και σε μία ελαφρώς διαφορετική μέθοδο αναπαράστασης της πληροφορίας που σχετίζεται με την διαδικτυακή κυκλοφορία. Παρακάτω περιγράφουμε τα δεδομένα κάθε είδους ξεχωριστά, εστιάζοντας στην σημασία των χαρακτηριστικών που τους αντιστοιχούν.

### 4.1.1 Τα Δεδομένα των Πακέτων

Μέσω του πρώτου είδους δεδομένων η δικτυακή κίνηση αναπαριστάνεται ως ένα σύνολο μεμονωμένων πακέτων μεταφοράς δεδομένων. Βάσει του μοντέλου TCP/IP τα πακέτα ορίζονται ως η βασική μονάδα μετάδοσης της πληροφορίας μεταξύ δικτύων. Είναι λογικό να υποθέσουμε ότι τα περισσότερα πακέτα τα οποία σχετίζονται με μία επίθεση θα διαφέρουν –ως προς τα διάφορα χαρακτηριστικά τους– από τα πακέτα εκείνα τα οποία αποτελούν μέρος της καλόβουλης κίνησης. Πάνω σε αυτήν την υπόθεση βασίζονται εξάλλου και τα μοντέλα Μηχανικής Μάθησης ούτως ώστε να «μάθουν» να ανιχνέυουν με επιτυχία τις επιθέσεις αυτές.

Μολονότι καλόβουλα και κακόβουλα πακέτα αξιοποιούνται συνδιαστικά ως προς την εκπαίδευση και την αξιολόγηση των μοντέλων, στην προκειμένη περίπτωση οι δύο κλάσεις πακέτων προέρχονται από δύο διαφορετικές πηγές: Το σύνολο καλόβουλων πακέτων [25] αναπαριστάνει την διαδικτυακή κυκλοφορία μεταξύ πανεπιστημιακών διακομιστών και διακομιστών DNS. Συγκεκριμένα χρησιμοποιούμε δύο ξεχωριστά σύνολα δεδομένων –τα πακέτα του ενός χρησιμοποιούνται στα πλαίσια της εκπαίδευσης των μοντέλων ενώ του δεύτερου αντίστοιχα στα πλαίσια της αξιολόγησης– κάθε ένα από τα οποία αντιστοιχεί και σε μία ώρα διαδικτυακής κίνησης περισσότερων από 4,000 μελών του πανεπιστημίου κατά την ημερομηνία της 24/04/2016. Αναφορικά με τα κακόβουλα πακέτα αυτά αποτελούν μέρος εφτά διαφορετικών DNS Amplification επιθέσεων, κάθε μία από τις οποίες έχει εκτελεστεί και μέσω ενός διαφορετικού Booter[1] υπό ελεγχόμενο περιβάλλον [26]. Αυτό συνεπάγεται ότι δύο πακέτα που παράγονται από δύο διαφορετικούς Booter ενδέχεται να παρουσιάζουν περισσότερες διαφορές μεταξύ τους από ότι δύο πακέτα που προέρχονται από τον ίδιο.

Παρόλο που τα σύνολα δεδομένων προέρχονται από διαφορετικές πηγές, τα χαρακτηριστικά μέσω των οποίων αναπαριστάνεται κάθε πακέτο παραμένουν τα ίδια, είτε το πακέτο αποτελεί μέρος καλόβουλης είτε κακόβουλης κίνησης. Εφόσον εξετάζουμε επιθέσεις οι οποίες πραγματοποιούνται μέσω διακομιστών DNS, το μεγαλύτερο μέρος των χαρακτηριστικών περιέχει πληροφορίες οι οποίες σχετίζονται με το αντίστοιχο πρωτόκολλο. Συγκεκριμένα κάθε πακέτο αποτελεί μέρος μίας DNS απάντησης σε ένα DNS ερώτημα το οποίο έχει εκτελεστεί είτε από τους διακομιστές του πανεπιστημίου, είτε μέσω των Booter στα πλαίσια της επίθεσης, αναλόγως της κλάσης του εκάστοτε πακέτου. Παρακάτω περιγράφουμε κάθε ένα από τα χαρακτηριστικά των πακέτων ξεχωριστά.[2,3]. Να τονίσουμε επιπλέον ότι παρακάτω χρησιμοποιείται το έντονο μαύρο χρώμα για να επισημάνουμε τα χαρακτηριστικά εκείνα τα οποία διατηρούνται κατά την φάση προεπεξεργασίας των συνόλων, συμβάλλοντας στην μετέπειτα εκπαίδευση/αξιολόγηση των μοντέλων,

[1] Ως Booter αναφερόμαστε στις επί πληρωμή υπηρεσίες που παρέχονται παράνομα στο διαδίκτυο με σκοπό την εκτέλεση DDoS επιθέσεων.

[2] Τα ονόματα των παρακάτω χαρακτηριστικών αντιστοιχούν στα ίδια ονόματα που χρησιμοποιούνται και εντός των αρχείων των δεδομένων.

[3] Ενδεικτικά αναφέρουμε ότι η πληροφορία που εμπεριέχεται σε κάθε πακέτο DNS διασπάται σε πέντε διαφορετικά τμήματα: Header, Question, Answer Resource Records, Authority Resource Records και Additional Resource Records, καθένα από τα οποία περιέχει πληροφορίες διαφορετικού είδους.

ενώ όλα τα υπόλοιπα χαρακτηριστικά αναφέρονται μονάχα για λόγους πληρότητας.

- frame.time.relative: Υποδηλώνει την χρονική στιγμή κατά την οποία το εκάστοτε πακέτο ελήφθη, ξεκινώντας την καταμέτρηση από μία αυθαίρετη χρονική στιγμή $t = 0$. Παρόλο που το χαρακτηριστικό αυτό καθαυτό δεν συμπεριλαμβάνεται στα τελικό σύνολο χαρακτηριστικών, χρησιμοποιείται ωστόσο ως προς την ταξινόμηση των πακέτων κατά την φάση συνένωσης καλόβουλων και κακόβουλων πακέτων σε ενιαία σύνολα. Κατά αυτόν τον τρόπο είναι εφικτό να προσομοιώσουμε μία πραγματική επίθεση η οποία λαμβάνει χώρα εξ απροόπτου, διαταράσσοντας τα φυσιολογικά μοτίβα κυκλοφορίας των διακομιστών. Εξάλλου όπως έχουμε ήδη αναφέρει η διάταξη των στοιχείων ενός συνόλου είναι σημαντική καθώς μέσω αυτής εκφράζεται η όποια χρονική πληροφορία.

- **ip.len**: Το μέγεθος του τμήματος IP του πακέτου σε byte.

- ip.src: Η διεύθυνση IP του αποστολέα του πακέτου, στην περίπτωσή μας η διεύθυνση ενός διακομιστή DNS.

- ip.dst: Η διεύθυνση IP του παραλήπτη του πακέτου, στην περίπτωσή μας η διεύθυνση ενός πανεπιστημιακού διακομιστή.

- **udp.length**: Το μέγεθος του τμήματος UDP του πακέτου σε byte.

- dns.flags.response: Ένα πεδίο ενός bit (τιμές 0 ή 1) το οποίο δηλώνει εάν το μήνυμα αποτελεί ένα DNS ερώτημα (DNS query - τιμή «0») ή μία DNS απάντηση (DNS response - τιμή «1»). Στην περίπτωσή μας κάθε πακέτο αποτελεί μέρος μίας DNS απάντησης, συνεπώς το χαρακτηριστικό αυτό μπορεί να αγνοηθεί.

- dns.flags.opcode: Ένα πεδίο τεσσάρων bit (τιμές 0-15) το οποίο δηλώνει το είδος του DNS ερωτήματος στο οποίο αντιστοιχεί ένα πακέτο:[4]

  - Τιμή «0» - STANDARD QUERY: Το ερώτημα είναι «κανονικό», δηλαδή ζητείται μία IP διεύθυνση βάσει ενός ονόματος τομέα.
  - Τιμή «1» - INVERSE QUERY: Το ερώτημα είναι «αντίστροφο», δηλαδή ζητείται το όνομα τομέα βάσει μίας IP διεύθυνσης.
  - Τιμή «2» - SERVER STATUS REQUEST: Το ερώτημα αποσκοπεί μονάχα στην εύρεση της κατάστασης του εκάστοτε διακομιστή.
  - Τιμές «3» έως «15» - RESERVED FOR FUTURE USE: Οι τιμές αυτές έχουν δεσμευτεί για μελλοντική χρήση.

  Η χρήση του χαρακτηριστικού αυτού δεν είναι απαραίτητη, καθώς ένα συντριπτικό ποστοστό των πακέτων που εξετάζουμε αφορά μονάχα «κανονικά» ερωτήματα.

[4]Μολονότι κάθε πακέτο αποτελεί τμήμα μίας DNS απάντησης, το χαρακτηριστικό αυτό αφορά το DNS ερώτημα στο οποίο αντιστοιχεί η εκάστοτε απάντηση.

- **dns.flags.authoritative**: Το πεδίο αυτό αποκτά νόημα μονάχα για τις DNS απαντήσεις. Πρόκειται για ένα πεδίο ενός bit το οποίο λαμβάνει την τιμή «1» όταν η τελική απάντηση στο ερώτημα έχει δοθεί από έναν Authoritative Name Server, δηλαδή από έναν διακομιστή DNS ο οποίος εξυπηρετεί συγκεκριμένα το όνομα τομέα του ερωτήματος και συνήθως ανήκει στην εταιρεία/οργανισμό υπο την ιδιοκτησία των οποίων βρίσκεται το εν λογω όνομα τομέα. Σε αυτή την περίπτωση είναι αρκετά πιθανό –αν και όχι πάντοτε– ότι το πακέτο αποτελεί μέρος καλόβουλης κίνησης.

- **dns.flags.truncated**: Ένα πεδίο ενός bit το οποίο λαμβάνει την τιμή «1» εάν το αρχικό μήνυμα έχει υποστεί περικοπή (truncation) λόγω μεγαλύτερου του επιτρεπόμενου μεγέθους που ορίζεται βάσει του πρωτόκολλου UDP. Σε αυτήν την περίπτωση επιχειρείται συνήθως η επαναποστολή του μηνύματος μέσω του πρωτόκολλου TCP. Στα πλαίσια εκτέλεσης μίας επίθεσης DDoS, το χαρακτηριστικό αυτό είναι πιθανό να κατέχει την τιμή «1», καθώς όπως έχουμε αναφέρει οι επιτιθέμενοι συνήθως προτιμούν να παράγουν ογκώδη μηνύματα.

- **dns.flags.recdesired**: Ένα πεδίο ενός bit το οποίο δηλώνει κατά πόσο η απάντηση του υποβεβλημένου ερωτήματος ζητήθηκε να ευρεθεί αναδρομικά, στην οποία περίπτωση λαμβάνει την τιμή «1». Γενικά υπάρχουν δύο μέθοδοι εύρεσης (lookup) της απάντησης σε ένα απλό DNS ερώτημα:

  – Iterative Lookup: Ο πελάτης εκτελεί το ερώτημα σε έναν διακομιστή DNS με σκοπό να λάβει την διεύθυνση IP του ζητούμενου ονόματος τομέα. Εάν η διεύθυνση αυτή δεν βρίσκεται αποθηκευμένη στην cache του, τότε ο διακομιστής απαντάει με την διεύθυνση ενός άλλου διακομιστή DNS ο οποίος ενδέχεται να κατέχει την διεύθυνση. Ο πελάτης επομένως εκτελεί επαναληπτικά κάμποσα ερωτήματα σε διαφορετικούς DNS διακομιστές, έως ότου ένας από αυτούς να του επιστρέψει την ζητούμενη διεύθυνση.

  – Recursive Lookup: Ο πελάτης και πάλι εκτελεί το ερώτημα σε έναν διακομιστή DNS με σκοπό να λάβει την διεύθυνση IP του ζητούμενου ονόματος τομέα. Εάν η διεύθυνση αυτή δεν βρίσκεται αποθηκευμένη στην cache του, τότε ο διακομιστής εκτελεί ο ίδιος ερωτήματα σε διαφορετικούς διακομιστές DNS έως ότου να λάβει την διέυθνση, την οποία με την σειρά του επιστρέφει πίσω στον πελάτη. Αυτή η μέθοδος είναι συνήθως γρηγορότερη, καθώς οι διακομιστές DNS που υποστηρίζουν ερωτήματα τέτοιου είδους αποθηκεύουν[5] στην cache τους την τελική απάντηση κάθε ερωτήματος που εκτελούν. Έχοντας εκτελέσει κάμποστα ερωτήματα, ένας τέτοιος διακομιστής είναι πλέον πιθανότερο να κατέχει τις απαντήσεις αρκετών –τουλάχιστον των δημοφιλέστερων– IP διευθύνσεων. Ωστόσο η μέθοδος αυτή είναι λιγότερο ασφαλής, καθώς η εκμετάλλευση της λειτουργίας τέτοιου είδους διακομιστών ως προς την εκτέλεση διαδικτυακών επιθέσεων δεν αποτελεί σπάνιο φαινόμενο.

[5]Προφανώς κάθε απάντηση δεν αποθηκεύεται επ αορίστου αλλά για ένα συγκεκριμένο χρονικό διάστημα γνωστό και ως Time-To-Live.

- **dns.flags.recavail**: Το πεδίο αυτό αποκτά νόημα μονάχα για τις DNS απαντήσεις. Πρόκειται για ενα πεδίο ενός bit το οποίο λαμβάνει την τιμή «1» όταν ο διακομιστής DNS στον οποίο υποβάλλεται ένα ερώτημα είναι αναδρομικός, δηλαδή υποστηρίζει την εκτέλεση αναδρομικών ερωτημάτων.

- **dns.flags.authenticated**: Το πεδίο αυτό επίσης αποκτά νόημα μονάχα για τις DNS απαντήσεις και σχετίζεται με το πρωτόκολλο DNSSEC, μία ανανεωμένη έκδοση του πρωτοκόλλου DNS με σκοπό την ασφαλέστερη επικοινωνία μεταξύ των πελατών και των διακομιστών DNS. Το πεδίο λαμβάνει την τιμή «1» μόνο όταν η απάντηση σε ένα ερώτημα είναι με μεγάλη βεβαιότητα αυθεντική, καθώς έχει επικυρωθεί μέσω του πρωτόκολλου DNSSEC.

- **dns.flags.checkdisable**: Το πεδίο αυτό επίσης σχετίζεται με το πρωτόκολλο DNSSEC. Πρόκειται για ενα πεδίο ενός bit το οποίο έχοντας λάβει την τιμή «1» ο πελάτης δηλώνει κατά την εκτέλεση ενός ερωτήματος ότι θα δεχτεί οποιαδήποτε απάντηση στο ερώτημά του, είτε αυτή είναι επικυρωμένη μέσω του πρωτόκολλου DNSSEC είτε όχι.

- dns.count.queries: Το πλήθος των καταχωρήσεων του τμήματος «Question» του DNS πακέτου, το οποίο περιέχει πληροφορίες σχετικά με το εκάστοτε ερώτημα που εκτελείται. Η τιμή αυτή είναι σχεδόν πάντοτε ίση με την μονάδα, γεγονός το οποίο σημαίνει ότι το εκάστοτε πακέτο αποτελεί τμήμα μίας και μόνο ερώτησης –ή αντίστοιχα απάντησης σε μία και μόνο ερώτηση. Συνεπώς το χαρακτηριστικό αυτό μπορεί να αφαιρεθεί από το τελικό σύνολο χαρακτηριστικών.

- **dns.count.answers**: Το πλήθος των καταχωρήσεων του τμήματος «Answer Resource Records» του DNS πακέτου. Το τμήμα αυτό περιέχει τις βασικές πληροφορίες μέσω των οποίων στην ουσία απαντάται ένα ερώτημα.

- **dns.count.auth_rr**: Το πλήθος των καταχωρήσεων του τμήματος «Authority Resource Records» του DNS πακέτου. Το τμήμα αυτό περιέχει κυρίως πληροφορίες αναφορικά με τους όποιους Authoritative διακομιστές DNS ενδέχεται να σχετίζονται με το εκάστοτε ερώτημα.

- **dns.count.add_rr**: Το πλήθος των καταχωρήσεων του τμήματος «Additional Resource Records» του DNS πακέτου. Το τμήμα αυτό περιέχει δευτερεύουσες πληροφορίες οι οποίες ενδέχεται να φανούν χρήσιμες σε ορισμένους πελάτες.

- dns.qry.name: Το πεδίο αυτό αναφέρεται στο όνομα τομέα του ερωτήματος, και συγκεκριμένα αντιστοιχεί στο πεδίο με τίτλο «Name» του τμήματος «Question» του DNS πακέτου.

- **dns.qry.type**: Το πεδίο αυτό αντιστοιχεί στο πεδίο «Type»[6] το οποίο βρίσκεται εντός του τμήματος «Question» του DNS πακέτου.

[6] Λίστα διαφορετικών τιμών του πεδίου «Type»: `https://en.wikipedia.org/wiki/List_of_DNS_record_types`, εξαιρουμένου του πίνακα με τίτλο «Obsolete record types».

### 4.1.2 Τα Δεδομένα των Ροών Πακέτων

Οι λεγόμενες ροές πακέτων αν και στην ουσία βασίζονται στην έννοια των πακέτων, παράλληλα εκφράζουν μία πιο γενική εικόνα της διαδικτυακής κυκλοφορίας που περιγράφουν. Συγκεκριμένα κάθε στοιχείο αυτού του τύπου αφορά στατιστικά τα οποία εξάγονται βάσει ενός δείγματος διαφορετικών πακέτων δεδομένων, τα οποία ωστόσο κατέχουν ορισμένα κοινά χαρακτηριστικά μεταξύ τους. Στην περίπτωσή μας τα χαρακτηριστικά αυτά είναι η διεύθυνση IP του αποστολέα και του παραλήπτη, οι θύρες αποστολής και παραλαβής, και τέλος το πρωτόκολλο (TCP ή UDP) το οποίο χρησιμοποιήθηκε για την μετάδοσή του πακέτου. Πακέτα τα οποία βρίσκονται χρονικά κοντά ενώ ταυτόχρονα μοιράζονται τα πέντε αυτά χαρακτηριστικά[7] ομαδοποιούνται έτσι ώστε να εξάγουμε ένα σύνολο χαρακτηριστικών, τα οποία εκφράζουν διαφόρων ειδών πληροφορίες, από το μέσο μέγεθος των πακέτων έως και τον μέγιστο χρόνο που μεσολαβεί μεταξύ της αποστολής δύο πακέτων εντός του ίδιου δείγματος.

Το σύνολο δεδομένων που χρησιμοποιούμε [27, 28] αφορά δώδεκα διαφορετικά είδη προσομοιωμένων DDoS επιθέσεων, οι οποίες διεξαχθεί ξεχωριστά η μία από την άλλη σε διάστημα μόλις δύο ημερών[8]. Συγκεκριμένα εφτά επιθέσεις εκτελέστηκαν την πρώτη μέρα, ενώ έντεκα την δεύτερη. Ωστόσο, καθώς μερικά είδη επιθέσεων πραγματοποιήθηκαν μονάχα την πρώτη ή την δεύτερη μέρα, κάνουμε λόγο για δώδεκα και όχι για δεκαοκτώ διαφορετικά είδη επιθέσεων:

- Επιθέσεις οι οποίες εκτελέστηκαν και τις δύο μέρες (6): LDAP, MSSQL, NetBIOS, UDP, Syn, UDPLag.
- Επιθέσεις οι οποίες εκτελέστηκαν αποκλειστικά την πρώτη μέρα (1): Portmap.
- Επιθέσεις οι οποίες εκτελέστηκαν αποκλειστικά την δεύτερη μέρα (5): DNS, NTP, SNMP, SSDP, TFTP.

Συνεπώς έχουμε στην διάθεσή μας δεκαοκτώ διαφορετικά σύνολα δεδομένων τα οποία περιέχουν ροές πακέτων που αντιστοιχούν τόσο σε καλόβουλη όσο και σε κακόβουλη διαδικτυακή κυκλοφορία. Αν και προφανές, να τονίσουμε ότι κάθε στοιχείο των παραπάνω συνόλων σχετίζεται αποκλειστικά είτε με καλόβουλη κίνηση είτε με την επίθεση που αντιστοιχεί στο σύνολο που το περιέχει, καθώς μία ροή πακέτων είναι αδύνατο να έχει προέλθει από ένα μικτό δείγμα καλόβουλων και κακόβουλων πακέτων[9]. Όσο αφορά τα χαρακτηριστικά που περιγράφουν τα στοιχεία των συνόλων, το πλήθος τους είναι αρκετά μεγαλύτερο από ότι ήταν στην περίπτωση των πακέτων. Συγκεκριμένα κάθε ροή πακέτων περιγράφεται από εβδομήντα έξι διαφορετικά χαρακτηριστικά. Παρόλα αυτά το τελικό σύνολο χαρακτηριστικών στο οποίο καταλήγουμε είναι πολύ μικρότερο. Πιο συγκεκριμένα:

[7] Να σημειωθεί ότι στην περίπτωση του συνόλου δεδομένων που εξετάζουμε, κάθε στοιχείο ροής πακέτων ενδέχεται να περιέχει πακέτα και των δύο κατευθύνσεων, δηλαδή αποστολής αλλά και παραλαβής.

[8] Οι δύο αυτές μέρες ωστόσο απέχουν αρκετά μεταξύ τους. Συγκεκριμένα οι επιθέσεις εκτελέστηκαν στις 12 Ιανουαρίου και 11 Μαρτίου.

[9] Αν και κάτι τέτοιο θα μπορούσε βέβαια θεωρητικά να συμβεί, στην περίπτωσή μας το φαινόμενο αυτό δεν συναντάται.

- Αφαιρούμε χαρακτηριστικά τα οποία κατείχαν μία και μόνο σταθερή τιμή για κάθε στοιχείο ανεξαρτήτως της κλάσης. Τα χαρακτηριστικά αυτά είναι τα εξής: Bwd PSH Flags, Fwd URG Flags, Bwd URG Flags, ECE Flag Count, FIN Flag Count, PSH Flag Count, Fwd Avg Bytes/Bulk, Fwd Avg Packets/Bulk, Fwd Avg Bulk Rate, Bwd Avg Bytes/Bulk, Bwd Avg Packets/Bulk, Bwd Avg Bulk Rate

- Αφαιρούμε χαρακτηριστικά τα οποία εκφράζουν την ίδια πληροφορία σε μεγάλο βαθμό. Το φαινόμενο αυτό γίνεται αντιληπτό τόσο από τα ονόματα ορισμένων χαρακτηριστικών (π.χ. Fwd Packet Length Mean και Avg Fwd Segment Size), όσο και μέσω του υπολογισμού των συντελεστών συσχέτισης Pearson ανάμεσα σε κάθε ζεύγος αριθμητικών μεταβλητών. Για παράδειγμα η τιμή του συντελεστή συσχέτισης μεταξύ των χαρακτηριστικών Fwd Packet Length Max και Fwd Packet Length Min ισούται με 0.996621.

- Αφαιρούμε χαρακτηριστικά η εξαγωγή των οποίων σε πραγματικό χρόνο αποτελεί υπό ρεαλιστικές συνθήκες μία περίπλοκη διαδικασία. Το σύνολο των χαρακτηριστικών Interarrival Time (IAT) αποτελεί ένα παράδειγμα των χαρακτηριστικών αυτών.

- Αφαιρούμε χαρακτηριστικά για τα οποία δεν μας παρέχονται συγκεκριμένες πληροφορίες ως προς την σημασία τους. Ένα παράδειγμα αποτελεί το σύνολο των χαρακτηριστικών Subflow.

- Τέλος αφαιρούμε ορισμένα χαρακτηριστικά τα οποία εξετάζοντάς τα θεωρήθηκε ότι δεν συμβάλλουν θετικά στην επίδοση των μοντέλων. Για παράδειγμα τα σύνολα χαρακτηριστικών Active και Idle κατέχουν την τιμή «0» για ένα συντριπτικό ποσοστό στοιχείων μεγαλύτερο του 80%.

Εφαρμόζοντας τα παραπάνω καταλήγουμε στην χρήση επτά μονάχα χαρακτηριστικών, τα οποία περιγράφονται παρακάτω:

- 1. Total Backward Packets: Το συνολικό πλήθος πακέτων παραλαβής.

- 2/3. Total Length of Fwd/Bwd Packets: Το συνολικό μέγεθος των πακέτων αποστολής/παραλαβής σε byte.

- 4/5. Fwd/Bwd Packet Length Min: Το ελάχιστο μέγεθος πακέτου μεταξύ των πακέτων αποστολής/παραλαβής σε byte.

- 6. URG Flag Count[10] Το πλήθος των πακέτων για τα οποία έχει οριστεί το URG bit.

- 7. act_data_pkt_fwd: Το πλήθος των πακέτων αποστολής τα οποία περιέχουν τουλάχιστον ένα byte πληροφορίας στο αντίστοιχο TCP Payload[11] τμήμα τους.

[10] Αν και στην περιγραφή του αναφέρεται ότι το χαρακτηριστικό αυτό εκφράζει ένα πλήθος, στην πράξη οι τιμές του είναι δυαδικές (0 ή 1). Για αυτόν τον λόγο αντιμετωπίζουμε το χαρακτηριστικό αυτό ως κατηγορική μεταβλητή.

[11] Το τμήμα αυτό διαχωρίζεται από την επικεφαλίδα (header), και ουσιαστικά εμπεριέχει την βασική πληροφορία του πακέτου.

## 4.2 Τα Μοντέλα

Σε αυτό το μέρος του κεφαλαίου παρουσιάζουμε τις αρχιτεκτονικές των μοντέλων που χρησιμοποιήσαμε για την διεξαγωγή των πειραμάτων. Με τον όρο «αρχιτεκτονική» αναφερόμαστε τόσο στις πληροφορίες οι οποίες αφορούν τα δομικά μέρη των μοντέλων (π.χ. το πλήθος επιπέδων/περιοχών), όσο και στις τιμές οποιονδήποτε άλλων παραμέτρων οι οποίες αφορούν την κατασκευή των μοντέλων. Επίσης θα πρέπει να τονίσουμε ότι προκειμένου να αποφευχθεί μετέπειτα οποιαδήποτε σύγχυση, αποδίδουμε σε κάθε μοντέλο και από ένα διαφορετικό όνομα. Τα ονόματα αυτά είναι σημαντικά καθώς πρόκειται να χρησιμοποιηθούν αργότερα κατά την παρουσίαση των αποτελεσμάτων, έτσι ώστε ο αναγνώστης να γνωρίζει την ακριβή αρχιτεκτονική των μοντέλων στα οποία αναφερόμαστε.

### 4.2.1 Τα Νευρωνικά Δίκτυα

Αρχικά θα εξετάσουμε τα μοντέλα των νευρωνικών δικτύων που χρησιμοποιούμε στα πλαίσια της εκτέλεσης των πειραμάτων. Πρωτού ξεκινήσουμε θα πρέπει να αναφέρουμε ωστόσο ότι αποφασίσαμε να αποφύγουμε την χρήση LSTM μονάδων, τόσο ως προς την κατασκευή ενός πλήρους LSTM δικτύου όσο και ως απλά συστατικά μέρη ενός μεγαλύτερου υβριδικού δικτύου, καθώς λόγω της απλότητας των χαρακτηριστικών τους κανένα από τα δύο είδη των δεδομένων στην περίπτωσή μας δεν χρήζει την εισαγωγή τέτοιων ισχυρών υπολογιστικών μονάδων, ενώ παράλληλα οι αρχιτεκτονικές που κατασκευάζουμε δεν θεωρούνται αρκετά βαθιές ώστε να αποτελέσουν πρόβλημα για την εκπαίδευση ενός RNN δικτύου. Εξάλλου βάσει ορισμένων πειραμάτων που εκτελέσαμε, οι επιδόσεις των LSTM δικτύων μπορούσαν κάλλιστα να επιτευχθούν μέσω RNN δικτύων αντίστοιχης αρχιτεκτονικής σε μικρότερο χρονικό διάστημα, καθώς το πλήθος των πράξεων που εκτελούν είναι σαφώς μικρότερο.

Το πρώτο δίκτυο που κατασκευάσαμε ονομάζεται «RNN-1», διότι το μοντέλο αυτό στην ουσία αποτελείται από ένα και μόνο κρυφό RNN επίπεδο διανύσματος κατάστασης εξιντατεσσάρων διαστάσεων, ενώ το επίπεδο εξόδου περιέχει μονάχα δύο νευρώνες, έναν για κάθε κλάση. Αναφορικά με την συνάρτηση ενεργοποίησης το κρυφό επίπεδο χρησιμοποιεί την συνάρτηση Rectified Linear Unit (ReLU), ενώ μετά το επίπεδο εξόδου δεν εφαρμόζεται καμία συνάρτηση[12]. Στον παρακάτω πίνακα παρουσιάζουμε την πλήρη αρχιτεκτονική αυτού του δικτύου. Με τον όρο $d$ συμβολίζουμε την διαστασιμότητα των εκάστοτε διανυσμάτων εισόδου.

Πίνακας 4.1: Η Αρχιτεκτονική του Δικτύου RNN-1.

| | **Είσοδος** | **RNN (w/ ReLU)** | **FC** |
|---|---|---|---|
| **Διαστασιμότητα** | $d$ | $d \to 64$ | $64 \to 2$ |

[12]Παρόλα αυτά κατά την διάρκεια της εκπαίδευσης χρησιμοποιείται η συνάρτηση Softmax στα πλαίσια του υπολογισμού του Cross Entropy Loss.

Η παραπάνω αρχιτεκτονική πρόκειται να εφαρμοστεί στην περίπτωση επίλυσης του προβλήματος της Ταξινόμησης με χρήση των ετικετών των δεδομένων, δηλαδή στα πλαίσια της Επιβλεπόμενης Μάθησης. Αντιστοίχως θα πρέπει να παρουσιάσουμε και την αρχιτεκτονική του δικτύου «AE-1», το οποίο χρησιμοποιούμε στην περίπτωση του προβλήματος της Ανίχνευσης Ανωμαλιών.

Πίνακας 4.2: Η Αρχιτεκτονική του Δικτύου AE-1.

| | **Είσοδος** | **FC (w/ ReLU)** | **FC** |
|---|---|---|---|
| **Διαστασιμότητα** | $d$ | $d \to \lceil d \cdot 0.8 \rceil$ | $\lceil d \cdot 0.8 \rceil \to d$ |

Από τον Πίνακα 4.2 διακρίνουμε ότι πρόκειται για έναν απλό αυτοκωδικοποιητή ενός μονάχα κρυφού επιπέδου, το οποίο δρα ως κωδικοποιητής και επίπεδο Bottleneck του δικτύου ταυτόχρονα, μειώνοντας την εκάστοτε διαστασιμότητα $d$ των δεδομένων κατά 20%. Στην συνέχεια το πλήθος των διαστάσεων των δεδομένων επιστρέφει και πάλι στην τιμή $d$ ούτως ώστε να πραγματοποιηθεί η ανακατασκευή της εισόδου στο επίπεδο εξόδου του δικτύου.

Όπως και στην περίπτωση του μοντέλου RNN-1 έτσι και τώρα παρατηρούμε μία αρκετά απλή αρχιτεκτονική, η οποία παρόλα αυτά αρκεί όσο αφορά τα δεδομένα βάσει των οποίων εργαζόμαστε αφού αυτά βρίσκονται εξαρχής σε έναν χώρο χαμηλών διαστάσεων. Επιπλέον θα πρέπει να τονίσουμε ότι η αρχιτεκτονική AE-1 δεν κάνει χρήση κάποιας RNN ή LSTM μονάδας, γεγονός το οποίο σημαίνει ότι το δίκτυο αυτό είναι ανίκανο να διαχειριστεί την χρονική πληροφορία των δεδομένων. Ωστόσο βάσει ορισμένων πειραμάτων που έχουν εκτελεστεί κατά την εκπόνηση αυτής της εργασίας, οι απλοί αυτοί αυτοκωδικοποιητές κατορθώνουν ίδιες –σε ορισμένες περιπτώσεις ακόμη και καλύτερες– επιδόσεις σε σχέση με αντίστοιχους αυτοκωδικοποιητές RNN. Εξάλλου, ακόμη και στην περίπτωση χρήσης των συστημάτων HTM στα πλαίσια της ανίχνευσης ανωμαλιών αποφεύγουμε την οργάνωση των δεδομένων σε χρονικές ακολουθίες, καθώς όπως έχουμε αναφέρει η πρακτική αυτή επηρεάζει την ορθή λειτουργία του συστήματος[13].

### 4.2.2 Τα Συστήματα HTM

Στην περίπτωση χρήσης των συστημάτων HTM ορίζουμε μία και μοναδική αρχιτεκτονική ανεξαρτήτως του προβλήματος προς επίλυση, καθώς για τα δύο είδη των δεδομένων, πακέτα και ροές πακέτων, η συγκεκριμένη αρχιτεκτονική φάνηκε να παράγει τα καλύτερα αποτελέσματα. Θα πρέπει να αναφέρουμε παρόλα αυτά ότι λόγω του συνδυασμού μεγάλου πλήθους παραμέτρων και υψηλού χρόνου εκπαίδευσης των συστημάτων HTM, είναι αδύνατο να προβούμε σε τεχνικές τύπου «Grid Search» με σκοπό τον υπολογισμό των βέλτιστων τιμών των παραμέτρων.

[13] Παρόλα αυτά θα πρέπει να αναφέρουμε πως ακόμη και εάν ένα σύστημα HTM δεν δέχεται τα δεδομένα οργανωμένα σε χρονικές ακολουθίες, δεν παύει ωστόσο να λαμβάνει υπόψη την χρονική πληροφορία μεταξύ δύο διαδοχικών στοιχείων. Στην περίπτωση των γραμμικών αυτοκωδικοποιητών από την άλλη, κάθε στοιχείο εισόδου θεωρείται απόλυτα ανεξάρτητο από το προηγούμενό του.

Η αναζήτηση τιμών συνεπώς πραγματοποιήθηκε εστιάζοντας την προσοχή μας σε ορισμένες παραμέτρους οι οποίες φάνηκαν να έχουν την μεγαλύτερη επίδραση στην επίδοση του μοντέλου. Οι βασικότερες αυτών είναι οι *columnDimensions, potentialRadius, localAreaDensity, activationThreshold, minThreshold*.

Σχετικά με την αρχιτεκτονική του μοντέλου θα πρέπει αρχικά να αναφέρουμε ότι χρησιμοποιούμε μία και μοναδικη HTM περιοχή, καθώς με εξαίρεση υπερβολικά περίπλοκων προβλημάτων (π.χ. Τεχνητή Όραση) μία περιοχή είναι αρκετή [29]. Η περιοχή αυτή αποτελείται από 1536 κύτταρα οργανωμένα σε ενενήντα έξι στήλες, δηλαδή κάθε στήλη περιέχει δεκαέξι κύτταρα. Επιπλέον, το γεγονός ότι το μοντέλο αποτελείται από μία και μόνο περιοχή σημαίνει ότι του αντιστοιχούν μονάχα ένα αντικείμενο «SpatialPooler» και ένα αντικείμενο «TemporalMemory» αντίστοιχα, οι παράμετροι των οποίων αποτελουν στην ουσία και τις παραμέτρους του συστήματος. Οι τιμές που χρησιμοποιήσαμε ως προς την κατασκευή του συστήματός μας παρουσιάζονται στους Πίνακες 4.3 και 4.4.

Πίνακας 4.3: Οι τιμές των παραμέτρων του αντικειμένου SpatialPooler του συστήματος HTM.

| **columnDimensions** | **potentialRadius** | **potentialPct** |
|---|---|---|
| 96 | $*$[14] | 0.9 |
| **globalInhibition** | **localAreaDensity** | **stimulusThreshold** |
| True | 0.20 | 1 |
| **synPermInactiveDec** | **synPermActiveInc** | **synPermConnected** |
| 0.01 | 0.05 | 0.5 |
| **minPctOverlapDutyCycle** | **DutyCyclePeriod** | **boostStrength** |
| 0.001 | 1000 | 2.0 |
| **wrapAround** | | |
| True | | |

Πίνακας 4.4: Οι τιμές των παραμέτρων του αντικειμένου TemporalMemory του συστήματος HTM.

| **columnDimensions** | **cellsPerColumn** | **activationThreshold** |
|---|---|---|
| 96 | 16 | 12 |
| **initialPermanence** | **connectedPermanence** | **minThreshold** |
| 0.20 | 0.50 | 6 |
| **maxNewSynapseCount** | **permanenceIncrement** | **permanenceDecrement** |
| 24 | 0.05 | 0.01 |
| **predictedSegmentDecrement** | **maxSegmentsPerCell** | **maxSynapsesPerSegment** |
| $*$[15] | 256 | 256 |

[14] Ορίζουμε την τιμή της παραμέτρου potentialRadius ως inputDimensions ÷ 64, έτσι ώστε το πλήθος των bit εισόδου με τα οποία ενδέχεται να είναι συνδεδεμένη κάθε στήλη της περιοχής να υπολογίζεται δυναμικά αναλόγως του εκάστοτε συνολικού τους πλήθους.

[15] Ορίζουμε την τιμή της παραμέτρου predictedSegmentDecrement ως localAreaDensity × permanenceIncrement καθώς η πρακτική αυτή προτείνεται από την Numenta.

Λόγω του ότι το πλήθος των παραμέτρων ενός συστήματος HTM είναι αρκετά μεγάλο, θα μπορούσαμε σε αυτό το σημείο να προτείνουμε μερικούς άτυπους κανόνες που χρησιμοποιήσαμε αναφορικά με την ρύθμιση των τιμών των παραμέτρων, οι οποίοι προέρχονται τόσο από την μελέτη της συμπεριφοράς των μοντέλων κατά την εκπόνηση της εργασίας όσο και από την ίδια την εταιρία Numenta:

- potentialRadius (SP): Η παράμετρος αυτή στην ουσία εκφράζει το πλήθος των bit εισόδου στο οποίο κάθε στήλη μίας περιοχης του συστήματος έχει πρόσβαση. Συνεπώς γίνεται εύκολα κατανοητό πως η τιμή αυτής της παραμέτρου είναι προτιμότερο να εξαρτάται άμεσα από το μήκος των εκάστοτε συμβολοσειρών εισόδου. Για παράδειγμα εμείς θέτουμε την τιμή αυτή ίση με την διαστασιμότητα της εισόδου διαρούμενη από μία παράμετρο $\lambda$, την οποία χειριζόμαστε ως μία νέα παράμετρο. Συγκεκριμένα για την περίπτωσή μας ορίζουμε $\lambda = 64$.
- globalInhibition (SP): Η παράμετρος αυτή είναι προτιμότερο να λαμβάνει την τιμή «Αληθής», ενεργοποιώντας κατά αυτόν τον τρόπο τον μηχανισμό ολικής συστολής, ο οποίος με την σειρά του οδηγεί στην μείωση του υπολογιστικού χρόνου του συστήματος διατηρώντας παράλληλα υψηλές επιδόσεις.
- localAreaDensity (SP): Η παράμετρος αυτή, η τιμή της οποίας θα πρέπει να ανήκει στο διάστημα $[0, 1]$, εκφράζει την επιθυμητή πυκνότητα των συμβολοσειρών εξόδου του αλγορίθμου Spatial Pooler, ένα μέγεθος το οποίο εναλλακτικά μπορεί να προσδιοριστεί μέσω της ακέραιας παραμέτρου numActiveColumnsPerInhArea. Παρόλα αυτά προτιμάται η χρήση της localAreaDensity καθώς η μέθοδος αυτή λαμβάνει υπόψη το μήκος των συμβολοσειρών εξόδου του αλγορίθμου. Σε αυτή την περίπτωση, η παράμετρος numActiveColumnsPerInhArea θα πρέπει να λαμβάνει την τιμή -1.
- stimulusThreshold (SP): Η τιμή αυτής της παραμέτρους θα μπορούσε να ορίστει αρκετά χαμηλά, ακόμα και ίση με την τιμή 0, εφόσον το τελικό σύνολο των ενεργοποιημένων στηλών κρίνεται ούτως ή άλλως κατά την εφαρμογή του μηχανισμού Συστολής.
- boostStrength (SP): Η παράμετρους αυτή θα πρέπει να λάβει μία θετική τιμή, ούτως ώστε να διατηρηθεί εν λειτουργία ο μηχανισμός Ενίσχυσης.
- synPermActiveInc/permanenceIncrement (SP/TP): Θα μπορούσαμε να αποδώσουμε την ίδια τιμή στις δύο αυτές παραμέτρους, καθώς και οι δύο αφορούν την τιμή αύξησης της μονιμότητας των συνάψεων. Την ίδια λογική μπορούμε να εφαρμόσουμε για τα ζευγάρια παραμέτρων synPermInactiveDec/permanenceDecrement και synPermConnected/connectedPermanence.
- maxSegmentsPerCell/maxSynapsesPerSegment (TP/TP): Αποδίδοντας αρκετά μεγάλες τιμές στις δύο αυτές παραμέτρους παρέχουμε στο σύστημα την ελευθερία να προσαρμόσει το μέγεθος των περιφερικών τμημάτων δενδρίτη καθώς και των συνάψεών του όπως αυτό κρίνει. Παρόλα αυτά θα πρέπει να είμαστε προσεκτικοί διότι καθώς αυξάνεται το πλήθος των συνάψεων παράλληλα αυξάνεται και ο χρόνος υπολογισμού της εξόδου του συστήματος.

## 4.3 Η Προετοιμασία των Πειραμάτων

Σε αυτό το μέρος περιγράφουμε οποιαδήποτε διαδικασία σχετίζεται με την προετοιμασία των πειραμάτων, πιο συγκεκριμένα την κατασκευή αλλά και την προεπεξεργασία των συνόλων εκπαίδευσης και αξιολόγησης που χρησιμοποιούνται στα πλαίσια της εκτέλεσής τους.

### 4.3.1 Κατασκευή και Προεπεξεργασία των Συνόλων Δεδομένων των Πακέτων

Στην περίπτωση χρήσης των δεδομένων των πακέτων έχουμε στην κατοχή μας συνολικά εννέα σύνολα δεδομένων. Τα δύο από αυτά, έστω Benign_1 και Benign_2, αντιστοιχούν σε πακέτα καλόβουλης διαδικτυακής κυκλοφορίας, ενώ τα υπόλοιπα εφτά, έστω Booter_1-7, εμπεριέχουν τα πακέτα εκείνα που σχετίζονται με τις επιθέσεις οι οποίες πραγματοποιήθηκαν μέσω των αντίστοιχων Booter. Στον Πίνακα 4.3 παρουσιάζεται το «Συνολικό Πλήθος Πακέτων» και το «Πλήθος Μοναδικών Πακέτων» κάθε ενός από τα παραπάνω σύνολα δεδομένων, αφότου βεβάιως έχουμε πρώτα αφαιρέσει τυχόν στοιχεία τα οποία περιέχουν ελλιπείς τιμές, καθώς επίσης και τα χαρακτηριστικά τα οποία δεν θεωρήσαμε σημαντικά για την επίλυση του προβλήματος[16].

Πίνακας 4.5: Το μέγεθος των συνόλων δεδομένων των πακέτων

| | **Συνολικό Πλήθος Πακ.** | **Πλήθος Μοναδικών Πακ.** |
|---|---|---|
| **Benign_1** | 950,717 | 12,297 |
| **Benign_2** | 716,631 | 10,664 |
| **Booter_1** | 10,657,318 | 447 |
| **Booter_2** | 7,878,084 | 13 |
| **Booter_3** | 8,806,934 | 6 |
| **Booter_4** | 13,069,305 | 83 |
| **Booter_5** | 2,038,373 | 100 |
| **Booter_6** | 3,535,088 | 588 |
| **Booter_7** | 5,991,235 | 309 |

Από τον παραπάνω πίνακα παρατηρούμε ότι παρόλο που το συνολικό πλήθος των καλόβουλων πακέτων είναι αρκετά μικρότερο από αυτό των κακόβουλων, το πλήθος των μοναδικών πακέτων είναι ωστόσο κατά έναν μεγάλο βαθμό μεγαλύτερο. Ιδιαίτερη εντύπωση κάνει για παράδειγμα το σύνολο δεδομένων «Booter_3» το οποίο εμπεριέχει μονάχα έξι ξεχωριστά πακέτα. Λόγω αυτής της ιδιαιτερότητας των δεδομένων, καθώς και ορισμένων χρονικών και χωρικών περιορισμών, τα σύνολα δεδομένων που πρόκειται να κατασκευάσουμε για την εκπαίδευση και αξιολόγηση των μοντέλων, θα περιέχουν ένα αρκετά μικρότερο ποσοστό των παραπάνω πακέτων. Επιπλέον όπως θα δούμε παρακάτω ορίζουμε το πλήθος των δεδομένων

[16] Βλέπε *Τα Σύνολα Δεδομένων/Τα Δεδομένα των Πακέτων*.

εκπαίδευσης έτσι ώστε αυτό να είναι αρκετά πιο μικρό από το αντίστοιχο πλήθος των δεδομένων αξιολόγησης, εφόσον ο μικρός αριθμός των μοναδικών στοιχείων μας επιτρέπει κάτι τέτοιο.

Όπως έχουμε ήδη προαναφέρει το πρόβλημα που λύνουμε ουσιαστικά κατατάσσεται στην κατηγορία των προβλημάτων Δυαδικής Ταξινόμησης. Στην περίπτωσή μας συγκεκριμένα, τις δύο διαφορετικές κλάσεις του προβλήματος αποτελούν η καλόβουλη και η κακόβουλη διαδικτυακή κυκλοφορία αντίστοιχα. Επιπλέον επιχειρούμε να αντιμετωπίσουμε το ίδιο πρόβλημα με δύο διαφορετικούς τρόπους, δηλαδή τόσο στα πλαίσια της Επιβλεπόμενης μάθησης κάνοντας χρήση των ετικετών, όσο και στα πλαίσια της Μη-Επιβλεπόμενης μάθησης όπου οι ετικέτες των δεδομένων που έχουμε στην διάθεσή μας χρησιμοποιούνται αποκλειστικά και μόνο για την αξιολόγηση των μοντέλων, και δεν συμβάλλουν στην εκπαίδευσή τους.

Σχετικά με την πρώτη προσέγγιση, δηλαδή αυτή της Επιβλεπόμενης Μάθησης, θα θέλαμε επιπλέον να εξακριβώσουμε κατά πόσο ένα μοντέλο είναι ικανό να ανιχνέυσει κακόβουλα πακέτα τα οποία σχετίζονται με μία επίθεση Booter_j, έχοντας εκπαιδευτεί με κακόβουλα πακέτα που αντιστοιχούν σε μία επίθεση Booter_i, όπου $i \neq j$. Ως προς την επίτευξη αυτού του σκοπού κατασκευάζουμε εφτά μικτά[17] σύνολα εκπαίδευσης και εφτά μικτά σύνολα αξιολόγησης. Κάθε ένα από τα σύνολα εκπαίδευσης περιέχει 30,000 στοιχεία τα οποία αποτελούνται από μία ίση αναλογία[18] καλόβουλων και κακόβουλων πακέτων. Κάθε $i$-οστό σύνολο εκπαίδευσης κατασκευάζεται συγχωνεύοντας τα δύο σύνολα δεδομένων Benign_1 και Booter_i, έπειτα ταξινομώντας τα πακέτα τους ως προς το χαρακτηριστικό frame.time.relative[19], και τέλος συλλέγοντας 30,000 από αυτά. Κατά τον ίδιο τρόπο παράγονται και τα σύνολα αξιολόγησης, με την διαφορά ότι τα καλόβουλα πακέτα προέρχονται από το σύνολο δεδομένων Benign_2, αλλά και το γεγονός ότι το μέγεθός τους ισούται με 500,000 αντί των 30,000 πακέτων, καθώς όπως έχουμε ήδη αναφέρει η ιδιαιτερότητα των δεδομένων μας επιτρέπει να κατασκευάσουμε σύνολα αξιολόγησης μεγέθους μεγαλύτερο από αυτό των συνόλων εκπαίδευσης.

Εφόσον έχουμε αναφερθεί στην διαδικασία κατασκευής των συνόλων δεδομένων που χρησιμοποιούνται για την περίπτωση της Επιβλεπόμενης Μάθησης, στην συνέχεια θα εξετάσουμε την δεύτερη και απλούστερη περίπτωση της Μη-Επιβλεπόμενης Μάθησης για την οποία διατηρούμε μονάχα ένα μη-μικτό σύνολο εκπαίδευσης αποτελούμενο από 30,000 πακέτα τα οποία προέρχονται αποκλειστικά και μόνο από το σύνολο δεδομένων Benign_1. Εφόσον στα πλαίσια αυτής της προσέγγισης κάθε κακόβουλο πακέτο θεωρείται ως μία «ανωμαλία» της συνηθισμένης

[17] Με τον όρο «μικτά» σύνολα δεδομένων αναφερόμαστε στα σύνολα δεδομένων τα οποία περιέχουν τόσο καλόβουλα όσο και κακόβουλα πακέτα.

[18] Τα ποσοστά των πακέτων κάθε κλάσης δεν θα διαφέρουν περισσότερο του 2%.

[19] Η ταξινόμηση των πακέτων ως προς την χρονική στιγμή έλευσης (frame.time.relative) είναι αναγκαία προϋπόθεση ούτως ώστε η διάταξη των στοιχείων των παραγόμενων συνόλων δεδομένων να προσομοιώνει όσο πιο πιστά γίνεται την δικτυακή κυκλοφορία ενός διακομιστή υπό επίθεση, αφού όπως έχουμε αναφέρει κάθε κλάση πακέτων προέρχεται από διαφορετική πηγή. Η διαδικασία αυτή κατέχει επιπλέον και πρακτική σημασία, καθώς τα πακέτα οργανώνονται σε ακολουθίες με σκοπό την αφομοίωση της χρονικής πληροφορίας. Είναι προφανές πως η μέθοδος αυτή θα ήταν ανώφελη –πολύ πιθανό και επιζήμια– για την επίδοση των μοντέλων εάν τα πακέτα ήταν διατεταγμένα με τυχαία σειρά.

διαδικτυακής κυκλοφορίας, τα πακέτα αυτά δεν χρειάζεται να συμπεριληφθούν στο σύνολο εκπαίδευσης. Παρόλα αυτά τα κακόβουλα πακέτα είναι απαραίτητα για την αξιολόγηση της επίδοσης των μοντέλων, για την οποία χρησιμοποιούμε τα ίδια εφτά μικτά σύνολα αξιολόγησης μεγέθους 500,000 πακέτων που χρησιμοποιούνται και στην περίπτωση της Επιβλεπόμενης Μάθησης. Με αυτόν τον τρόπο μπορούμε να συγκρίνουμε άμεσα τις δύο μεθόδους εκπαίδευσης μεταξύ τους.

Σχετικά με την προεπεξεργασία των επιμέρους τιμών των δεδομένων, στην περίπτωση εφαρμογής των νευρωνικών δικτύων η μοναδική πρακτική που ακολουθούμε είναι η τυποποίησή τους. Η περίπτωση χρήσης των συστημάτων HTM από την άλλη χρήζει ολικής κωδικοποιησης των χαρακτηριστικών ούτως ώστε κάθε στοιχείο να μετατραπεί σε μία SDR συμβολοσειρά. Στον Πίνακα 4.6 παρουσιάζονται οι παράμετροι της εν λόγω κωδικοποίησης. Όπως φαίνεται και από τον πίνακα, το μήκος των κωδικοποιημένων συμβολοσειρών ισούται συνολικά με 6,433 bit, ενώ η πυκνότητά τους δεν ξεπερνάει το 10%. Αυτό ουσιαστικά σημαίνει πως για κάθε παραγόμενη συμβολοσειρά, το ποσοστό των ενεργών bit της συμβολοσειράς θα αποτελεί πάντοτε μικρότερο του 10% του συνολικού μήκους της συμβολοσειράς, συγκεκριμένα 8.89%.

Πίνακας 4.6: Η κωδικοποίηση των χαρακτηριστικών των πακέτων ως μία ενιαία SDR συμβολοσειρά.

| **Χαρακτηριστικό** | **Σύνολο Πιθανών Τιμών** | **Διάστημα Κωδικοποίησης**[20] | $n$ | $w$ | **Πυκνότητα** |
|---|---|---|---|---|---|
| ip.len | $[45, 1500]$ | $[45, 1500]$ | 1024 | 31 | 3.03% |
| udp.length | $[25, 5405]$ | $[25, 5000]$ | 2048 | 63 | 3.08% |
| dns.flags.* | $\{0, 1\}$ | - | 126 | 63 | 50.00% |
| dns.count.answers | $[0, 50650]$ | $[0, 300]$ | 512 | 23 | 4.49% |
| dns.count.auth_rr | $[0, 61673]$ | $[0, 50]$ | 256 | 23 | 8.98% |
| dns.count.add_rr | $[0, 24320]$ | $[0, 50]$ | 256 | 23 | 8.98% |
| dns.qry.type | -[21] | - | 1581 | 31 | 1.96% |
| **Τελική SDR Συμβολοσειρά** | - | - | 6433 | 572 | 8.89% |

Εκτός ίσως από το γεγονός ότι η Numenta προτείνει μία μικρή τιμή πυκνότητας, δεν φαίνεται να υπάρχει κάποιος άλλος κανόνας κωδικοποίησης των δεδομένων. Η διαδικασία αυτή θα πρέπει να πραγματοποιείται μονάχα αφότου έχουμε πρώτα μελετήσει τις τιμές των χαρακτηριστικών των δεδομένων. Ωστόσο ακόμη και αυτή η θεωρητική προσέγγιση ενδέχεται να αποδειχθεί ανεπαρκής, αναγκάζοντάς μας να πειραματιστούμε με διαφορετικές τιμές των παραμέτρων έως ότου να καταλήξουμε σε μία αναπαράσταση της εισόδου, ικανή να βοηθήσει το εκάστοτε σύστημα ώστε να κατορθώσει υψηλές επιδόσεις.

[20] Η στήλη αυτή ισχύει αποκλειστικά για τα αριθμητικά χαρακτηριστικά και στην ουσία αφορά τις τιμές των παραμέτρων minval και maxval του αντικειμένου *ScalarEncoder*, βλέπε *Η Βιβλιοθήκη NuPIC/Η Κλάση ScalarEncoder*.

[21] Για το σύνολο πιθανών τιμών του χαρακτηριστικού *dns.qry.type* βλέπε *Τα Σύνολα Δεδομένων/Τα Δεδομένα Πακέτων*.

### 4.3.2 Κατασκευή και Προεπεξεργασία των Συνόλων Δεδομένων των Ροών Πακέτων

Αναφορικά με τα δεδομένα των ροών πακέτων υπενθυμίζουμε ότι έχουμε στην διάθεσή μας δεκαοκτω διαφορετικά σύνολα δεδομένων τα οποία αντιστοιχούν συνολικά σε δώδεκα διαφορετικές επιθέσεις, με κάθε σύνολο να περιέχει τόσο καλόβουλα όσο και κακόβουλα στοιχεία ροών πακέτων. Λόγω του υπερβολικά μεγάλου μεγέθους ορισμένων από τα σύνολα αυτά, αλλά και του μικρού πλήθους των καλόβουλων δεδομένων, χρησιμοποιούμε ένα σαφώς μικρότερο αλλά παράλληλα ικανοποιητικά μεγάλο ποσοστό των δεδομένων ως προς την εκπαίδευση αλλά και την αξιολόγηση των μοντέλων. Παρακάτω περιγράφουμε τον τρόπο κατασκευής των συνόλων που συμμετέχουν στις διαδικασίες αυτές. Υπενθυμίζουμε πως σκοπός μας είναι ναι μεν να εξακριβώσουμε εάν τα μοντέλα μπορούν να διαχωρίσουν την καλόβουλη από την κακόβουλη δικτυακή κίνηση, αλλά παράλληλα θα θέλαμε επίσης να υπολογίσουμε τον βαθμό στον οποίο τα μοντέλα είναι ικανά να ανιχνεύσουν επιθέσεις τις οποίες δεν έχουν συναντήσει κατά την εκπαίδευση.

Στην περίπτωση της Επιβλεπόμενης Μάθησης το σύνολο εκπαίδευσης που κατασκευάζουμε εμπεριέχει δεδομένα τα οποία προέρχονται από έξι διαφορετικές επιθέσεις[22]. Το σύνολο αυτό αποτελείται συνολικά από 173,368 στοιχεία, ενώ τα επιμέρους ποσοστά κάθε είδους κίνησης παρουσιάζονται λεπτομερέστερα στον Πίνακα 4.7. Ωστόσο θα πρέπει να αναφέρουμε ότι η αναλογία 40/60 των καλόβουλων/κακόβουλων στοιχείων εντός του συνόλου εκπαίδευσης έχει επιτευχθεί τεχνητά δημιουργώντας μέχρι και πέντε επιπλέον αντίγραφα για κάθε καλόβουλο στοιχείο, και εντάσσοντάς τα στο σύνολο εκπαίδευσης. Η πρακτική αυτή εφαρμόζεται καθώς το αρχικό πλήθος των καλόβουλων στοιχείων είναι σημαντικά μικρότερο των κακόβουλων, γεγονός το οποίο είναι ικανό να επηρεάσει αρνητικά την εκπαίδευση των μοντέλων. Παρόλα αυτά η περίπτωση της Μη-Επιβλεπόμενης Μάθησης δεν πάσχει από τέτοιου είδους προβλήματα καθώς η εκπαίδευση πραγματοποιείται ούτως ή άλλως κάνοντας χρήση των στοιχείων της κλάσης της καλόβουλης κίνησης και μόνο. Επομένως το αντίστοιχο σύνολο εκπαίδευσης για αυτήν την περίπτωση εμπεριέχει μονάχα τα 11,547 καλόβουλα στοιχείων που συναντάμε στο αντίστοιχο σύνολο εκπαίδευσης του Πίνακα 4.7.

Πίνακας 4.7: Το σύνολο δεδομένων των ροών πακέτων εκπαίδευσης για την περίπτωση της Επιβλεπόμενης Μάθησης.

| | **Καλόβουλα** | **Κακόβουλα** | | | | | |
|---|---|---|---|---|---|---|---|
| | | **LDAP** | **MSSQL** | **NetBIOS** | **SYN** | **UDP** | **UDPLag** |
| **Πλήθος** | 68,187 | 17,543 | 17,541 | 17,476 | 17,540 | 17,539 | 17,542 |
| **Ποσοστό** | 39.33% | 10.11% | 10.11% | 10.08% | 10.11% | 10.11% | 10.11% |

Σε αντίθεση με την προηγούμενη περίπτωση των δεδομένων των πακέτων –στα πλαίσια της οποίας η εκπαίδευση των δικτύων αποδείχτηκε ευκολότερη υπόθεση– στην περίπτωση των ροών πακέτων κατασκευάζουμε επιπλέον δύο σύνολα επικύρωσης τα οποία χρησιμοποιούμε κατά την εκπαίδευση των νευρωνικών δικτύων. Ο

[22]LDAP, MSSQL, NetBIOS, SYN, UDP και UDPLag.

τρόπος κατασκευής των συνόλων επικύρωσης είναι αντίστοιχος με αυτόν των συνόλων εκπαίδευσης, με την μόνη διαφορά ότι το μέγεθός τους είναι σαφώς μικρότερο. Το πρώτο σύνολο επικύρωσης αφορά την περίπτωση Επιβλεπόμενης Μάθησης και αποτελείται από 25,000 στοιχεία συνολικά, τα επιμέρους ποσοστά των οποίων παρουσιάζονται στον Πίνακα 4.8. Από την άλλη το σύνολο επικύρωσης που κατασκευάζεται για την περίπτωση της Μη-Επιβλεπόμενης Μάθησης αποτελείται από 6,678 ροές πακέτων οι οποίες αποτελούν μέρος της καλόβουλης κίνησης.

Πίνακας 4.8: Το σύνολο δεδομένων των ροών πακέτων επικύρωσης για την περίπτωση της Επιβλεπόμενης Μάθησης.

| | **Καλόβουλα** | **Κακόβουλα** | | | | | |
|---|---|---|---|---|---|---|---|
| | | **LDAP** | **MSSQL** | **NetBIOS** | **SYN** | **UDP** | **UDPLag** |
| **Πλήθος** | 3,584 | 3,584 | 3,570 | 3,571 | 3,566 | 3,571 | 3,571 |
| **Ποσοστό** | 14.33% | 14.26% | 14.28% | 14.28% | 14.26% | 14.28% | 14.28% |

Απομένει μονάχα να κατασκευάσουμε το σύνολο αξιολόγησης, το οποίο θα θέλαμε να περιέχει δεδομένα σχετικά και με τις δώδεκα επιθέσεις. Παρατηρούμε πως οι έξι επιθέσεις οι οποίες συμμετέχουν στην κατασκευή των συνόλων εκπαίδευσης και επικύρωσης είναι και οι μόνες που εκτελέστηκαν και τις δύο ημέρες, γεγονός το οποίο σημαίνει ότι σε κάθε μία από αυτές αντιστοιχεί ένα σαφώς μεγαλύτερο πλήθος δεδομένων. Συνεπώς ως προς την κατασκευή του συνόλου αξιολόγησης μπορούμε ελεύθερα να αντλήσουμε δεδομένα σχετικά με τις έξι παραπάνω επιθέσεις, αρκεί μονάχα να στραφούμε προς τις ροές πακέτων που αφορούν την πρώτη ημέρα εκτέλεσης των έξι αυτών επιθέσεων ούτως ώστε να εξασφαλίσουμε ότι το σύνολο αξιολόγησης δεν θα μοιράζεται κοινά στοιχεία με κανένα από τα δύο σύνολα δεδομένων των Πινάκων 4.7 και 4.8[23]. Όσο αφορά τις υπόλοιπες έξι επιθέσεις[24] οι οποίες εκτελέστηκαν μονάχα την μία από τις δύο μέρες, τα στοιχεία τους μπορούν να ενταχθούν άφοβα εφόσον δεν περιέχονται στα σύνολα δεδομένων εκπαίδευσης/επικύρωσης. Το τελικό σύνολο αξιολόγησης –αποτελούμενο από 5,696,426 στοιχεία συνολικά– παρουσιάζεται αναλυτικότερα στον Πίνακα 4.9.

Πίνακας 4.9: Το σύνολο δεδομένων των ροών πακέτων αξιολόγησης.

| | **Καλόβουλα** | **Κακόβουλα** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **DNS** | **LDAP** | **MSSQL** | **NTP** | **NetBIOS** | **Portmap** | **SNMP** | **SSDP** | **Syn** | **TFTP** | **UDP** | **UDPLag** |
| **Πλήθος** | 70,068 | 495,931 | 491,686 | 497,716 | 472,604 | 499,492 | 186,960 | 498,336 | 499,139 | 499,335 | 497,285 | 495,963 | 491,911 |
| **Ποσοστό** | 1.23% | 8.71% | 8.63% | 8.74% | 8.30% | 8.77% | 3.28% | 8.75% | 8.76% | 8.77% | 8.73% | 8.71% | 8.64% |

Τέλος δεν θα πρέπει να παραλείψουμε να αναφέρουμε ότι τα στοιχεία που συνθέτουν όλα τα παραπάνω σύνολα δεδομένων είναι σε κάθε περίπτωση ταξινομημένα ως προς τον χρόνο άφιξής τους, ο οποίος εκφράζεται μέσω του χαρακτηριστικού *Timestamp*[25]. Ειδικά για τα μικτά σύνολα δεδομένων αξίζει επίσης να τονίσουμε πως τα στοιχεία τους ομαδοποιούνται σε χρονικές ομάδες βάσει των διαφορετικών

[23]Τα σύνολα εκπαίδευσης και επικύρωσης εμπεριέχουν ροές πακέτων οι οποίες προέρχονται από την δεύτερη μέρα εκτέλεσης των έξι επιθέσεων.

[24]DNS, NTP, SNMP, SSDP, TFTP και Portmap.

[25]Το χαρακτηριστικό αυτό αφαιρείται αφότου ταξινομήσουμε τα στοιχεία, όπως ακριβώς συνέβη και στην περίπτωση των πακέτων με το χαρακτηριστικό *frame.time.relative*.

επιθέσεων, κάτι το οποίο συμβαίνει διότι κάθε επίθεση έχει εκτελεστεί εντός μίας διαφορετικής ώρας της εκάστοτε ημέρας. Το γεγονός αυτό εξασφαλίζει ότι τα κακόβουλα στοιχεία κάθε χρονικής ακολουθίας που κατασκευάζεται θα αντιστοιχούν αποκλειστικά σε μία και μόνο επίθεση, κάτι το οποίο επιθυμούμε καθώς η ταυτόχρονη εκτέλεση δύο διαφορετικών επιθέσεων DDoS μοιάζει μάλλον απίθανη.

Όσο αφορά την προεπεξεργασία των τιμών των δεδομένων, η διαδικασία αυτή και πάλι διαφοροποιείται αναλόγως του μοντέλου. Στην απλούστερη περίπτωση των νευρωνικών δικτύων εφαρμόζουμε «0-1 Κανονικοποίηση» των τιμών των αριθμητικών χαρακτηριστικών –σε αντίθεση με την περίπτωση των δεδομένων των πακέτων για τα οποία χρησιμοποιήσαμε την μέθοδο της Τυποποίηση»– καθώς η μέθοδος αυτή πολύ απλά οδηγεί σε καλύτερα αποτελέσματα σε συνδυασμό με τα στοιχεία των ροών πακέτων. Σχετικά με την κωδικοποίηση των δεδομένων σε SDR συμβολοσειρές, οι τιμές των παραμέτρων των συναρτήσεων κωδικοποίησης που χρησιμοποιούμε παρουσιάζονται στον Πίνακα 4.10.

Πίνακας 4.10: Η κωδικοποίηση των χαρακτηριστικών των ροών πακέτων ως μία ενιαία SDR συμβολοσειρά.

| **Χαρακτηριστικό** | **Σύνολο Πιθανών Τιμών**[26] | **Διάστημα Κωδικοποίησης** | $n$ | $w$ | **Πυκνότητα** |
|---|---|---|---|---|---|
| Total Backward Packets | $[0, 817]$ | $[0, 10000]$ | 1024 | 55 | 5.37% |
| Total Length of Fwd Packets | $[0, 69310]$ | $[0, 800]$ | 256 | 31 | 12.10% |
| Total Length of Bwd Packets | $[0, 1468048]$ | $[0, 5000]$ | 1024 | 55 | 5.37% |
| Fwd Packet Length Min | $[0, 1715]$ | $[0, 1700]$ | 512 | 43 | 8.39% |
| Bwd Packet Length Min | $[0, 1072]$ | $[0, 1700]$ | 512 | 43 | 8.39% |
| URG Flag Count | $\{0, 1\}$ | - | 86 | 43 | 50.00% |
| act_data_pkt_fwd | $[0, 167]$ | $[0, 150]$ | 128 | 23 | 17.96% |
| **Τελική SDR Συμβολοσειρά** | - | - | 3542 | 293 | 8.27% |

Μερικές από τις παραπάνω επιλογές τιμών ίσως να φανούν περίεργες στον αναγνώστη. Για παράδειγμα ενώ η μέγιστη τιμή του χαρακτηριστικού *Total Bwd Packets* ισούται με μόλις 817, εμείς χρησιμοποιούμε ένα άνω όριο ίσο με 10,000 ως προς την κωδικοποίησή του, καθώς η πρακτική αυτή απλά φαίνεται να οδηγεί σε καλύτερα αποτελέσματα σε σχέση με την περίπτωση χρήσης ενός άνω ορίου το οποίο ανταποκρίνεται στην πραγματικότητα, π.χ. μία τιμή ίση με 1,000. Συνεπώς παρατηρούμε ότι μία «λογική» προσέγγιση επιλογής των παραμέτρων κωδικοποίησης δεν αποτελεί πάντοτε την βέλτιστη επιλογή.

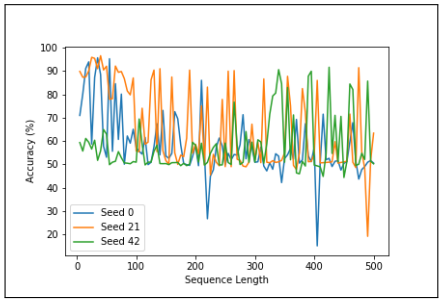
[26] Θα πρέπει να τονίσουμε ότι η στήλη αυτή εκφράζει μονάχα το σύνολο τιμών που συναντάμε εντός του συνόλου εκπαίδευσης.

### 4.3.3 Οργάνωση των Δεδομένων σε Χρονικές Ακολουθίες

Το τελευταίο βήμα της προεπεξεργασίας των δεδομένων αποτελεί η οργάνωσή τους σε χρονικές ακολουθίες. Να τονίσουμε εδώ ότι αναλόγως του είδους του μοντέλου που χρησιμοποιούμε –νευρωνικά δίκτυα ή συστηματα HTM– ακολουθούμε και μία ελαφρώς διαφορετική προσέγγιση. Στην περίπτωση των συστημάτων HTM επιλέγουμε να οργανώσουμε τα στοιχεία σε ακολουθίες βάσει ενός χρονικού διαστήματος $dt$, το οποίο μετράται σε δευτερόλεπτα. Για παράδειγμα η πρώτη ακολουθία κατασκευάζεται βάσει όλων των στοιχείων που συναντάμε μεταξύ των χρονικών στιγμών $t_0 = 0$ και $t_1 = t_0 + dt$. Αντιστοίχως, η δεύτερη ακολουθία κατασκευάζεται κατά τον ίδιο τρόπο βάσει των χρονικών στιγμών $t_1$ και $t_2 = t_1 + dt$, και ούτω καθεξής έως ότου κάθε στοιχείο του εκάστοτε συνόλου δεδομένων να ανήκει και σε μία ακολουθία.

Βάσει αυτής της προσέγγισης είναι προφανές ότι οι παραγόμενες ακολουθίες δεν κατέχουν κάποιο σταθερό μήκος. Ωστόσο η μέθοδος αυτή προτιμάται καθώς όπως φανερώνει και το σχήμα στα δεξιά είναι αδύνατο να διακρίνουμε κάποια τιμή σταθερού μήκους των ακολουθιών ικανή να ευνοήσει το μοντέλο. Συγκεκριμένα στο Σχήμα 4.1 παρουσιάζεται η πορεία της ακρίβειας τριών αρχιτεκτονικά όμοιων[27] συστημάτων HTM –τα οποία έχουν εκπαιδευτεί βάσει του ίδιου συνόλου δεδομένων– ως προς την ακρίβεια που επιτυγχάνουν πάνω στο ίδιο σύνολο αξιολόγησης. Από το γράφημα αυτό γίνεται αντιληπτό όχι μόνο ότι το ίδιο μήκος ακολουθιών επιδρά με αρκετα διαφορετικό τρόπο στην επίδοση κάθε μοντέλου, αλλά παράλληλα και ότι τα ίδια μοντέλα ενδέχεται να αντιδράσουν με εντελώς διαφορετικό τρόπο σε δύο ξεχωριστές τιμές μήκους, ακόμη και αν αυτές βρίσκονται αριθμητικά κοντά μεταξύ τους.

Σχήμα 4.1: Το Γράφημα της Ακρίβειας τριών αρχιτεκτονικά όμοιων συστημάτων HTM διαφορετικού seed, ως προς την τιμή σταθερού μήκους των ακολουθιών των πακέτων




Για αυτόν τον λόγο προτιμάμε την διαδικασία κατασκευής ακολουθιών που περιγράφουμε στην πρώτη παράγραφο. Εξάλλου ένα πλεονέκτημα των συστημάτων HTM είναι η ικανότητά τους να διαχειρίζονται ακολουθίες μεταβλητού μήκους. Όσο αφορά τώρα τις τιμές των χρονικών διαστημάτων $dt$ που χρησιμοποιούμε, για την περίπτωση των δεδομένων των πακέτων θέτουμε την τιμή $dt$ ίση με 0.05, ενώ για την περίπτωση των ροών πακέτων ορίζουμε $dt = 0.5$, καθώς τα δεδομένα αυτά λόγω της φύσης τους τείνουν να καταφτάνουν με αργότερους ρυθμούς.

[27] Παρά το γεγονός ότι τα τρία συστήματα μοιράζονται την ίδια αρχιτεκτονική, διαφέρουν ωστόσο ως προς την εκτέλεση των στοχαστικών τους διαδικασιών, κάτι το οποίο επιτυγχάνεται εύκολα ορίζοντας διαφορετικές τιμές του seed.

Για την περίπτωση των νευρωνικων δικτύων από την άλλη επιλέγουμε την κατασκευή ακολουθιών σταθερού μήκους, αφενός επειδή η διαχείριση ακολουθιών μεταβλητού μήκους αποτελεί μία ελαφρώς δυσκολότερη διαδικασία από τα μοντέλα αυτά, αφετέρου διότι το μήκος των ακολουθιών δεν φαίνεται να επηρεάζει τα δίκτυα στον βαθμό που επηρεάζει τα συστήματα HTM. Συγκεκριμένα και στις δύο περιπτώσεις δεδομένων, πακέτων και ροών πακέτων, κατασκευάζουμε χρονικές ακολουθίες σταθερού μήκους πενήντα στοιχείων.

Τέλος θα πρέπει να τονίσουμε ότι η οργάνωση των στοιχείων κάθε συνόλου σε χρονικές ακολουθίες αποκτά νόημα μονάχα για την περίπτωση της επίλυσης του προβλήματος στα πλαίσια της Επιβλεπόμενης Μάθησης, κάτι το οποίο ισχυεί και για τα δύο είδη μοντέλων. Όσο αφορά τα νευρωνικά δίκτυα υπενθυμίζουμε ότι για την περίπτωση της Μη-Επιβλεπόμενης Μάθησης εφαρμόζουμε το μοντέλο AE-1, το οποίο λόγω της ίδιας της αρχιτεκτονικής του καθίσταται ανίκανο να επεξεργαστεί οποιουδήποτε είδους χρονική δομή βάσει της οποίας ενδέχεται τα δεδομένα να έχουν οργανωθεί. Αναφορικά με τα συστήματα HTM, στο τρίτο κεφάλαιο της εργασίας έχουμε αναφέρει τον λόγο για τον οποίο τα δεδομένα δεν θα πρέπει να οργανώνονται σε χρονικές ακολουθίες στα πλαίσια της ανίχνευσης ανωμαλιών, καθώς η πρακτική αυτή οδηγεί το μοντέλο στην εκτέλεση λανθασμένων προβλέψεων.

## 4.4 Περιγραφή των Πειραμάτων

Έχοντας αναφερθεί σε κάθε πτυχή της διαδικασιας προετοιμασίας των πειραμάτων μπορούμε πλέον να πρχωρήσουμε στην περιγραφή τους. Τα πειράματα αυτά μπορούν να κατηγοριοποιηθούν βάσει δύο κριτηρίων: του είδους των δεδομένων (πακέτα ή ροές πακέτων) και της προσέγγισης λύσης του προβλήματος (Επιβλεπόμενη ή Μη-Επιβλεπόμενη Μάθηση). Θα ξεκινήσουμε παρέχοντας μία συνοπτική περιγραφή των πειραμάτων τα οποία αφορούν τα δεδομένα των πακέτων:

- Πείραμα Επιβλεπόμενης Μάθησης: Το εκάστοτε μοντέλο (RNN-1 ή HTM) εκπαιδεύεται εφτά φορές, κάνοντας κάθε φορά χρήση μονάχα ενός από τα εφτά μικτά σύνολα εκπαίδευσης μεγέθους 30,000 πακέτων, ενώ παράλληλα η επίδοσή του μετριέται βάσει και των εφτά διαφορετικών μικτών συνόλων αξιολόγησης μεγέθους 500,000 πακέτων. Μέσω αυτής της διαδικασίας μπορούμε να εξακριβώσουμε κατά πόσο είναι εφικτό για ένα μοντέλο να διαχωρίσει τα καλόβουλα από τα κακόβουλα πακέτα, αλλά ταυτόχρονα και να γενικέυσει την γνώση του πάνω σε κακόβουλα πακέτα τα οποία σχετίζονται με ελαφρώς διαφορετικές επιθέσεις από αυτήν βάσει της οποίας το μοντέλο εκπαιδεύτηκε. Υπενθυμίζουμε ότι κάθε ένα από τα παραπάνω μικτά σύνολα δεδομένων περιέχει ίση αναλογία καλόβουλων και κακόβουλων πακέτων.
- Πείραμα Μη-Επιβλεπόμενης Μάθησης: Το εκάστοτε μοντέλο (AE-1 ή HTM) εκπαιδεύεται μία φορά βάσει ενός μη-μικτού συνόλου εκπαίδευσης μεγέθους 30,000 πακέτων, ενώ στην συνέχεια η επίδοσή του μετριέται βάσει των παραπάνω εφτά μικτών συνόλων αξιολόγησης.

Αντιστοίχως παρακάτω παρουσιάζουμε τα πειράματα τα οποία σχετίζονται με τα δεδομένα των ροών πακέτων:

- Πείραμα Επιβλεπόμενης Μάθησης: Το εκάστοτε μοντέλο (RNN-1 ή HTM) εκπαιδεύεται βάσει ενός μικτού συνόλου εκπαίδευσης μεγέθους 173,368 στοιχείων το οποίο περιέχει καλόβουλα αλλά και κακόβουλα δεδομένα που αντιστοιχούν σε έξι διαφορετικές διαδικτυακές επιθέσεις[28]. Στην συνέχεια η επίδοση του μοντέλου αξιολογείται βάσει ενός μικτού συνόλου δεδομένων αποτελούμενο συνολικά από 5,696,426 καλόβουλα και κακόβουλα στοιχεία τα οποία σχετίζονται με δώδεκα διαφορετικές επιθέσεις[29]. Βάσει αυτού του πειράματος επιχειρούμε να διαπιστώσουμε εάν το μοντέλο είναι ικανό να διαχωρίσει τις δύο κλάσεις διαδικτυακής κυκλοφορίας μεταξύ τους, καθώς επίσης και να εξετάσουμε τον βαθμό στον οποίο η επαφή του μοντέλου με ένα πλήθος διαφορετικών επιθέσεων μπορεί να συμβάλλει στην ανίχνευση νέων, άγνωστων επιθέσεων διαφορετικού είδους[30]. Υπενθυμίζουμε ότι οι αναλογίες των στοιχείων των δύο παραπάνω μικτών συνόλων δεδομένων δεν είναι ίσες, και παρουσιάζονται αναλυτικά στους πίνακες 4.7 και 4.9 αντίστοιχα.

- Πείραμα Μη-Επιβλεπόμενης Μάθησης: Το εκάστοτε μοντέλο (AE-1 ή HTM) εκπαιδεύεται βάσει ενός μη-μικτού συνόλου εκπαίδευσης μεγέθους 11,547 στοιχείων τα οποία αποτελούν μέρος της καλόβουλης κίνησης, ενώ στην συνέχεια η επίδοσή του μετριέται βάσει του παραπάνω μικτού συνόλου αξιολόγησης.

Προτού προχωρήσουμε στην παρουσίαση των αποτελεσμάτων των πειραμάτων, θα πρέπει πρώτα να αναφερθούμε σύντομα στην μετρική που χρησιμοποιούμε ως προς την αξιολόγηση των μοντέλων. Στα πλαίσια της ανίχνευσης της κακόβουλης κυκλοφορίας, και γενικότερα στα πλαίσια της επίλυσης προβλημάτων Δυαδικής Ταξινόμησης, συνηθίζεται να χρησιμοποιούμε τους εξής τέσσερις όρους:

- True Positive (TP): Τα στοιχεία τα οποία αποτελούν μέρος κακόβουλης κίνησης, και τα οποία ταξινομήθηκαν ορθά από το μοντέλο ως μέρος αυτής.

- False Positive (FP): Τα στοιχεία τα οποία μολονότι αποτελούν μέρος καλόβουλης κίνησης, ταξινομήθηκαν από το μοντέλο ως κακόβουλα.

- True Negative (TN): Τα στοιχεία τα οποία αποτελούν μέρος καλόβουλης κίνησης, και τα οποία ταξινομήθηκαν ορθά από το μοντέλο ως μέρος αυτής.

- False Negative (FN): Τα στοιχεία τα οποία μολονότι αποτελούν μέρος κακόβουλης κίνησης, ταξινομήθηκαν από το μοντέλο ως καλόβουλα.

[28] LDAP, MSSQL, NetBIOS, SYN, UDP και UDPLag.

[29] Οι επιθέσεις αυτές αφορούν τις έξι επιθέσεις, τα στοιχεία των οποίων συναντάμε εντός του σύνολο εκπαίδευσης, καθώς επίσης και τις επιθέσεις DNS, NTP, SNMP, SSDP, TFTP και Portmap.

[30] Αν και οι επιθέσεις ανήκουν σε διαφορετικές υποκατηγορίες, όλες τους ωστόσο ανήκουν στην ευρύτερη κατηγορία των επιθέσεων DDoS.

Προφανώς επιθυμούμε το μοντέλο μας να επιτυγχάνει υψηλό αριθμό των True Negative και True Positive, ενώ αντίστοιχα θα θέλαμε ο αριθμός των False Negative και False Positive να είναι χαμηλός. Συνδυάζοντας τις παραπάνω τέσσερις τιμές μπορούμε να ορίσουμε ένα πλήθος διαφορετικών μετρικών[31], κάθε μία από τις οποίες προσφέρει και από έναν διαφορετικό τρόπο ερμηνείας των αποτελεσμάτων. Στην περίπτωσή μας ωστόσο θα χρησιμοποιήσουμε τις απλές μετρικές «True Positive Rate» (TPR) και «True Negative Rate» (TNR), οι οποίες δεν εκφράζουν τίποτε άλλο παρά μόνο το συνολικό ποσοστό των κακόβουλων και καλόβουλων στοιχείων αντίστοιχα που το μοντέλο ταξινόμησε επιτυχώς. Συγκεκριμένα ισχύει:

$$\bullet\ \mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \qquad \bullet\ \mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

Όσο αφορά τα πειράματα που σχετίζονται με τα δεδομένα των ροών πακέτων, στα πλαίσια των οποίων το αντίστοιχο σύνολο αξιολόγησης περιέχει δεδομένα προερχόμενα από δώδεκα διαφορετικές επιθέσεις ως μέρος της ίδιας κλάσης της κακόβουλης διαδικτυακής κυκλοφορίας, η μετρική TPR υπολογίζεται για κάθε επίθεση ξεχωριστά. Συνεπώς θα μπορούσαμε να ορίσουμε τις τιμές $\mathrm{TPR}_j$, όπου $j \in \{1, 2, ..., 12\}$, ως εξής:

$$\mathrm{TPR}_j = \frac{\mathrm{TP}_j}{\mathrm{TP}_j + \mathrm{FN}_j}$$

Στα πλαίσια του παραπάνω ορισμού οι τιμές $\mathrm{TP}_j$ εκφράζουν το πλήθος των κακόβουλων στοιχείων που σχετίζονται με την επίθεση $j$ τα οποία ταξινομήθηκαν επιτυχώς από το μοντέλο. Αντιστοίχως οι τιμές $\mathrm{FN}_j$ αποτελούν το πλήθος των κακόβουλων στοιχείων της επίθεσης $j$ τα οποία ταξινομήθηκαν λανθασμένα ως μέρος της καλόβουλης κίνησης. Εφαρμόζοντας αυτή την πρακτική γίνεται επομένως εφικτό να διακρίνουμε ποιές επιθέσεις μπορούν να ανιχνευθούν εύκολα από το μοντέλο, αλλά και ποιές από αυτές του προκαλούν σύγχυση, οδηγώντας το στην ταξινόμησή των στοιχείων τους ως μέρος της συνηθισμένης καλόβουλης κίνησης.

## 4.5 Τα Αποτελέσματα των Πειραμάτων

Στο τελευταίο μέρος αυτού του κεφαλαίου παρουσιάζουμε τα αποτελέσματα κάθε πειράματος που εκτελέστηκε στα πλαίσια της εκπόνησης αυτής της εργασίας, ξεκινώντας από τα πειράματα τα οποία αφορούν τα δεδομενα των πακέτων, και συνεχίζοντας με τα αντίστοιχα πειράματα των ροών πακέτων. Πριν προχωρήσουμε στην παρουσίαση των αποτελεσμάτων θα πρέπει να αναφέρουμε ότι κάθε ξεχωριστό πείραμα έχει εκτελεστεί τρεις φορές βάσει τριών αρχιτεκτονικά όμοιων μοντέλων, τα οποία ωστόσο διαφέρουν ως προς την εκτέλεση των στοχαστικών τους διαδικασιών[32]. Συνεπώς κάθε τιμή που υπολογίζεται βάσει των μετρικών TPR και TNR

[31] Όπως για παράδειγμα η μετρική Precision, Recall, F1, κ.λπ.

[32] Η εκπαίδευση των δύο είδών μοντέλων εμπεριέχει κάμποσες στοχαστικές διαδικασίες, όπως για παράδειγμα η σειρά τροφοδότησης των δεδομένων σε αυτά, η αρχικοποίηση των συναπτικών βαρών στην περίπτωση των νευρωνικών δικτύων, η αρχικοποίηση των τιμών μονιμότητας των συνάψεων στην περίπτωση των συστημάτων HTM, κ.λπ.

αποτελεί στην ουσία τον μέσο όρο των επιμέρους τιμών που εξάγουμε μέσω των τριών μοντέλων. Επιπλέον υπολογίζουμε την τυπική απόκλιση των τιμών αυτών, παρέχοντας έτσι μία ένδειξη ως προς τον βαθμό συνέπειας των αποτελεσμάτων.

### 4.5.1 Τα Αποτελέσματα των Πειραμάτων βάσει των Δεδομένων των Πακέτων

Ξεκινώντας από τα πειράματα που εκτελέστηκαν στα πλαίσια της Επιβλεπόμενης Μάθησης, υπενθυμίζουμε ότι κάθε μοντέλο εκπαιδεύεται βάσει εφτά διαφορετικών συνόλων εκπαίδευσης τα οποία διαφέρουν ως προς τα κακόβουλα δεδομένα που περιέχουν, καθώς κάθε ένα από τα σύνολα αυτά αντιστοιχεί και σε έναν από τους εφτά διαφορετικούς Booter. Αντίστοιχα τα μοντέλα αξιολογούνται ως προς εφτά διαφορετικά σύνολα αξιολόγησης τα οποία ακολουθούν το ίδιο μοτίβο. Οι σαράντα εννέα διαφορετικές τιμές TPR που προέκυψαν από την εκτέλεση όλων των διαφορετικών συνδυασμών εκπαίδευσης-αξιολόγησης βάσει του μοντέλου RNN-1 παρουσιάζονται αναλυτικά στον Πίνακα 4.11.

Πίνακας 4.11: Οι τιμές TPR (%) που επιτυγχάνει το μοντέλο RNN-1 ως προς τους σαράντα εννέα διαφορετικούς συνδυασμούς εκπαίδευσης-αξιολόγησης, βάσει των δεδομένων των πακέτων στα πλαίσια της Επιβλεπόμενης Μάθησης.

| Εκπ. \ Αξ. | **Booter 1** | **Booter 2** | **Booter 3** |
|---|---|---|---|
| **Booter 1** | 100.00% (± 0.00) | 100.00% (± 0.00) | 100.00% (± 0.00) |
| **Booter 2** | 100.00% (± 0.00) | 99.99% (± 0.01) | 99.98% (± 0.02) |
| **Booter 3** | 100.00% (± 0.00) | 99.98% (± 0.02) | 99.98% (± 0.03) |
| **Booter 4** | 0.00% (± 0.00) | 0.00% (± 0.00) | 0.00% (± 0.00) |
| **Booter 5** | 0.00% (± 0.00) | 0.00% (± 0.00) | 0.00% (± 0.00) |
| **Booter 6** | 100.00% (± 0.00) | 99.99% (± 0.01) | 99.98% (± 0.01) |
| **Booter 7** | 100.00% (± 0.00) | 100.00% (± 0.00) | 100.00% (± 0.00) |

| Εκπ. \ Αξ. | **Booter 4** | **Booter 5** | **Booter 6** |
|---|---|---|---|
| **Booter 1** | 0.39% (± 0.22) | 0.31% (± 0.08) | 100.00% (± 0.00) |
| **Booter 2** | 0.02% (± 0.01) | 0.08% (± 0.06) | 99.98% (± 0.01) |
| **Booter 3** | 0.02% (± 0.02) | 0.10% (± 0.08) | 99.98% (± 0.02) |
| **Booter 4** | 99.75% (± 0.01) | 66.33% (± 9.64) | 0.00% (± 0.00) |
| **Booter 5** | 3.14% (± 0.39) | 99.61% (± 0.03) | 0.00% (± 0.00) |
| **Booter 6** | 0.00% (± 0.00) | 0.07% (± 0.05) | 99.98% (± 0.01) |
| **Booter 7** | 0.00% (± 0.00) | 0.07% (± 0.05) | 100.00% (± 0.00) |

| Εκπ. \ Αξ. | **Booter 7** | | |
|---|---|---|---|
| **Booter 1** | 99.99% (± 0.00) | | |
| **Booter 2** | 99.94% (± 0.05) | | |
| **Booter 3** | 99.93% (± 0.06) | | |
| **Booter 4** | 0.00% (± 0.00) | | |
| **Booter 5** | 0.00% (± 0.00) | | |
| **Booter 6** | 99.95% (± 0.03) | | |
| **Booter 7** | 100.00% (± 0.00) | | |

Τα αντίστοιχα αποτελέσματα που επιτεύχθηκαν μέσω του συστήματος HTM παρουσιάζονται στον Πίνακα 4.12.

Πίνακας 4.12: Οι τιμές TPR (%) που επιτυγχάνει το σύστημα HTM ως προς τους σαράντα εννέα διαφορετικούς συνδυασμούς εκπαίδευσης-αξιολόγησης, βάσει των δεδομένων των πακέτων στα πλαίσια της Επιβλεπόμενης Μάθησης.

| Εκπ. \ Αξ. | **Booter 1** | **Booter 2** | **Booter 3** |
|---|---|---|---|
| **Booter 1** | 99.99% (± 0.01) | 99.96% (± 0.02) | 99.97% (± 0.01) |
| **Booter 2** | 99.96% (± 0.02) | 99.97% (± 0.02) | 99.98% (± 0.01) |
| **Booter 3** | 99.60% (± 0.52) | 99.99% (± 0.00) | 100.00% (± 0.00) |
| **Booter 4** | 17.09% (± 0.05) | 6.91% (± 0.00) | 34.86% (± 0.00) |
| **Booter 5** | 0.00% (± 0.00) | 0.00% (± 0.00) | 0.00% (± 0.00) |
| **Booter 6** | 99.99% (± 0.01) | 99.99% (± 0.01) | 100.00% (± 0.00) |
| **Booter 7** | 99.94% (± 0.06) | 99.98% (± 0.03) | 99.96% (± 0.03) |

| Εκπ. \ Αξ. | **Booter 4** | **Booter 5** | **Booter 6** |
|---|---|---|---|
| **Booter 1** | 0.46% (± 0.12) | 0.01% (± 0.00) | 99.88% (± 0.06) |
| **Booter 2** | 0.03% (± 0.00) | 0.03% (± 0.03) | 99.79% (± 0.11) |
| **Booter 3** | 0.15% (± 0.14) | 0.00% (± 0.00) | 99.82% (± 0.11) |
| **Booter 4** | 90.19% (± 0.05) | 0.00% (± 0.00) | 1.54% (± 1.81) |
| **Booter 5** | 0.07% (± 0.05) | 25.55% (± 30.86) | 0.00% (± 0.00) |
| **Booter 6** | 90.05% (± 0.01) | 0.04% (± 0.03) | 100.00% (± 0.00) |
| **Booter 7** | 90.05% (± 0.05) | 0.07% (± 0.05) | 99.64% (± 0.51) |

| Εκπ. \ Αξ. | **Booter 7** | | |
|---|---|---|---|
| **Booter 1** | 99.82% (± 0.06) | | |
| **Booter 2** | 99.73% (± 0.11) | | |
| **Booter 3** | 99.81% (± 0.12) | | |
| **Booter 4** | 1.36% (± 1.87) | | |
| **Booter 5** | 0.00% (± 0.00) | | |
| **Booter 6** | 100.00% (± 0.00) | | |
| **Booter 7** | 99.65% (± 0.48) | | |

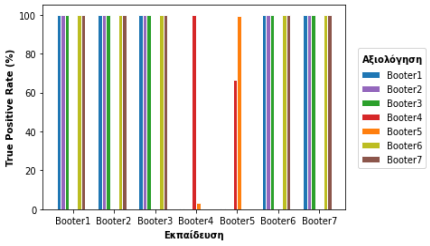
Τόσο στην περίπτωση του δικτύου RNN-1 όσο και στην αντίστοιχη περίπτωση του συστήματος HTM λαμβάνουμε αρκετά ικανοποιητικά αποτελέσματα. Το βασικό πρόβλημα που προκύπτει έγκειται στην αδυναμία των μοντέλων να ανιχνεύσουν τα κακόβουλα πακέτα που προέρχονται από τους Booter_4 και Booter_5. Αν και η ερμηνεία των αποτελεσμάτων δεν αποτελεί μία εύκολη υπόθεση, αξίζει ωστόσο να τονίσουμε ότι εντός της εργασίας [26] αναφέρεται πως οι επιθέσεις που πραγματοποιήθηκαν μέσω των Booter_1, Booter_2 και Booter_3 μοιράζονται το ίδιο όνομα τομέα όσο αφορά το DNS ερώτημά τους, κάτι το οποίο ενδέχεται να σχετίζεται με το γεγονός ότι τα μεγέθη των πακέτων των τριών αυτών επιθέσεων ακολουθούν παρόμοιες κατανομές. Το ίδιο φαίνεται να ισχύει αντίστοιχα και για τις επιθέσεις που εκτελέστηκαν μέσω των Booter_6 και Booter_7, με αποτέλεσμα οι επιθέσεις Booter_4 και Booter_5 να είναι οι μόνες οι οποίες δεν σχετίζονται με κάποια άλλη επίθεση[33].

[33] Στην ίδια εργασία επίσης αναφέρεται ότι η επίθεση του Booter 5 πρακτικά θεωρείται αποτυχημένη καθώς ο ρυθμός επίθεσης μόλις αγγίζει τα 6.11Mbps.

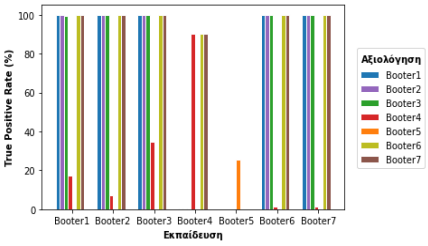
Ωστόσο το σύστημα HTM φαίνεται να υπερτερεί του μοντέλου RNN-1 όσο αφορά την ανίχνευση της επίθεσης Booter_4, τουλάχιστον στις περιπτώσεις για τις οποίες η εκπαίδευση των δύο μοντέλων πραγματοποιείται βάσει των συνόλων δεδομένων Booter_1, 2 και 3. Το φαινόμενο αυτό διακρίνεται καλύτερα στο Σχήμα 4.2. Αντιθέτως, στην περίπτωση που η εκπαίδευση βασίζεται στους Booters_4 και 5 το μοντέλο RNN-1 μοιάζει πιο ικανό να ανιχνεύσει τις δύο αυτές επιθέσεις, ενώ όταν τα δεδομένα εκπαίδευσης προέρχονται από τους Booters_6 και 7, και τα δύο μοντέλα παρουσιάζουν πλήρη αδυναμία ανίχνευσης των επιθέσεων Booter_4 και 5. Το σημαντικότερο σημείο που θα πρέπει να τονίσουμε αφορά ωστόσο την ανίχνευση των επιθέσεων Booter_6 και Booter_7 στην περίπτωση εκπαίδευσης βάσει του συνόλου δεδομένων Booter_4, για την οποία το σύστημα HTM σαφώς και υπερτερεί του μοντέλου RNN-1, καταφέρνοντας να ανιχνεύσει μέχρι και το 90% των κακόβουλων πακέτων.

Σχήμα 4.2: Γραφική αναπαράσταση των αποτελεσμάτων των Πινάκων 4.11 και 4.12.

(α′) RNN-1




(β′) Σύστημα HTM




Στην συνέχεια θα συγκρίνουμε τα δύο μοντέλα ως προς την μετρική TNR, δηλαδή ως προς το ποσοστό των καλόβουλων πακέτων που ταξινομούν επιτυχώς. Συγκεκριμένα στον Πίνακα 4.13 παρουσιάζονται τα ποσοστά TNR που κατορθώνονται από τα μοντέλα RNN-1 και HTM ως προς τα εφτά διαφορετικά σύνολα εκπαίδευσης βάσει των οποίων έχουν εκπαιδευτεί.

Πίνακας 4.13: Οι τιμές TNR (%) που επιτυγχάνουν τα δύο μοντέλα RNN-1 και HTM ως προς τα εφτά διαφορετικά σύνολα δεδομένων των πακέτων εκπαίδευσης στα πλαίσια της Επιβλεπόμενης Μάθησης.

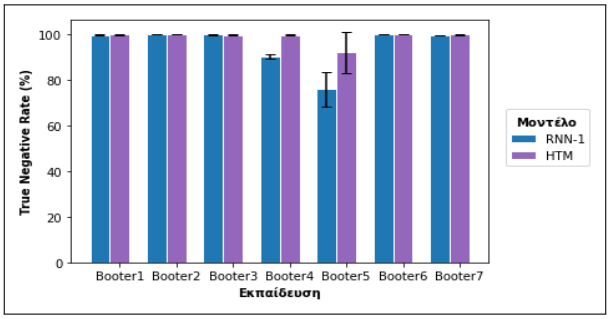
| Μοντέλο \ Εκπ. | **Booter 1** | **Booter 2** | **Booter 3** | **Booter 4** | **Booter 5** | **Booter 6** | **Booter 7** |
|---|---|---|---|---|---|---|---|
| **RNN-1** | 99.85% (± 0.05) | 99.95% (± 0.01) | 99.89% (± 0.03) | 90.23% (± 1.09) | 75.93% (± 7.71) | 99.91% (± 0.02) | 99.86% (± 0.02) |
| **HTM** | 99.90% (± 0.07) | 99.94% (± 0.02) | 99.85% (± 0.07) | 99.78% (± 0.13) | 92.24% (± 9.01) | 99.99% (± 0.00) | 99.93% (± 0.08) |

Από τον παραπάνω πίνακα παρατηρούμε ότι η εκπαίδευση των μοντέλων βάσει των συνόλων εκείνων που περιέχουν κακόβουλα πακέτα σχετικά με τις επιθέσεις Booter_4 και Booter_5 επηρεάζει αρνητικά την ικανότητά τους να αναγνωρίζουν τα καλόβουλα πακέτα ως μέρος της συνηθισμένης καλόβουλης διαδικτυακής κυκλοφορίας. Συνδυάζοντας αυτή την πληροφορία με το γεγονός ότι τα μοντέλα δυσκολεύονται να ανιχνεύσουν τα εν λόγω κακόβουλα πακέτα όταν έχουν εκ-

παιδευτεί βάσει δεδομένων που σχετίζονται με διαφορετικές επιθέσεις, μπορούμε να ισχυριστούμε ότι οι Booter_4 και Booter_5 κάνουν μια αρκετά καλή δουλειά ως προς την μεταμφίεση των επιθέσεων ως μέρος καλόβουλης κυκλοφορίας, αν και όπως έχουμε σημειώσει προηγουμένως η επίθεση που εκτελείται μέσω του Booter_5 πρακτικά θεωρείται αποτυχημένη [26].

Όσο αφορά συγκεκριμένα την σύγκριση των μοντέλων RNN-1 και HTM μεταξύ τους, είναι ξεκάθαρο πως η αρνητική επιρροή που ασκούν τα δεδομένα των συνόλων Booter_4 και Booter_5 πάνω στα δύο μοντέλα είναι ισχυρότερη στην περίπτωση του νευρωνικού δικτύου, καθώς το σύστημα HTM τείνει να ταξινομεί λανθασμένα ένα μικρότερο ποσοστό των κακόβουλων πακέτων σε σχέση με το μοντέλο RNN-1. Παρόλα αυτά κρίνοντας από την τυπική απόκλιση των τιμών TNR η επίθεση Booter_5 δεν παύει να αποπροσανατολίζει και τα δύο είδη μοντέλων, οδηγώντας αρχιτεκτονικά όμοια μοντέλα σε αρκετά διαφορετικά αποτελέσματα. Οι παρατηρήσεις αυτές απεικονίζονται επίσης γραφικά μέσω του Σχήματος 4.3.

Σχήμα 4.3: Γραφική αναπαράσταση των αποτελεσμάτων του Πίνακα 4.13.




Τέλος απομένει μονάχα να εξετάσουμε τα αποτελέσματα των πειραμάτων που εκτελέστηκαν στα πλαίσια της Μη-Επιβλεπόμενης Μάθησης. Το πείραμα αυτό αφορά την εκπαίδευση των μοντέλων AE-1 και HTM χρησιμοποιώντας ένα σύνολο δεδομένων αποτελούμενο από 30,000 καλόβουλα πακέτα, ενώ η επίδοσή τους αξιολογείται βάσει των ίδιων εφτά μικτών συνόλων δεδομένων που χρησιμοποιήθηκαν και για την περίπτωση της Επιβλεπόμενης Μάθησης. Όσο αφορά την ταξινόμηση των καλόβουλων πακέτων ο αυτοκωδικοποιητής AE-1 επιτυγχάνει ένα πολύ υψηλό ποσοστό TNR ίσο με 99.83%($\pm$0.01), ξεπερνώντας το αντίστοιχο ποσοστό που κατορθώνει το σύστημα HTM, το οποίο ισούται με 89.46%($\pm$1.12). Την υπεροχή του δικτύου AE-1 παρατηρούμε και στην περίπτωση της ταξινόμησης των κακόβουλων πακέτων, όπως εξάλλου φαίνεται και από τον Πίνακα 4.14.

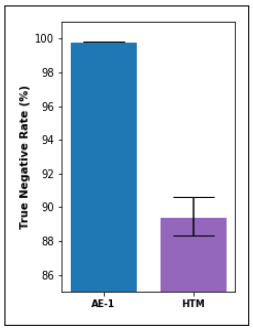
Πίνακας 4.14: Οι τιμές TPR (%) που επιτυγχάνουν τα δύο μοντέλα AE-1 και HTM ως προς τα εφτά διαφορετικά σύνολα δεδομένων των πακέτων αξιολόγησης στα πλαίσια της Μη-Επιβλεπόμενης Μάθησης.

| Σύνολο Αξ. / Μοντέλο | **Booter 1** | **Booter 2** | **Booter 3** | **Booter 4** | **Booter 5** | **Booter 6** | **Booter 7** |
|---|---|---|---|---|---|---|---|
| **AE-1** | 100.00% ($\pm$ 0.00) | 100.00% ($\pm$ 0.00) | 100.00% ($\pm$ 0.00) | 0.85% ($\pm$ 0.00) | 4.53% ($\pm$ 0.02) | 100.00% ($\pm$ 0.00) | 100.00% ($\pm$ 0.00) |
| **HTM** | 83.24% ($\pm$ 0.00) | 82.58% ($\pm$ 0.00) | 80.48% ($\pm$ 0.00) | 90.46% ($\pm$ 0.12) | 0.31% ($\pm$ 0.04) | 92.42% ($\pm$ 0.00) | 91.23% ($\pm$ 0.00) |

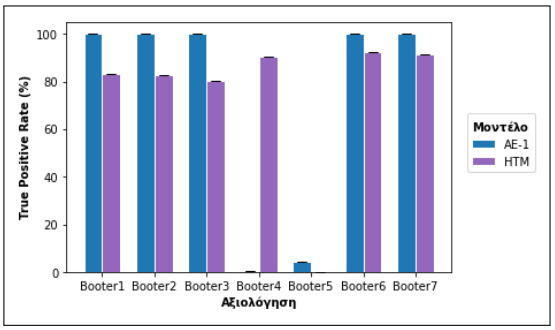
Τα παραπάνω αποτελέσματα παρουσιάζονται γραφικά μέσω του Σχήματος 4.4. Παρατηρούμε ότι αναφορικά με τις επιθέσεις των Booter_1, 2, 3, 6 και 7 το δίκτυο AE-1 ταξινομεί σωστά το 100% των αντίστοιχων κακόβουλων πακέτων, σε αντίθεση με το σύστημα HTM το οποίο ταξινομεί λανθασμένα μέχρι και ένα ποσοστό της τάξης του 20%. Παρόλα αυτά η υπεροχή του συστήματος HTM έναντι του δικτύου AE-1 στην περίπτωση της επίθεσης Booter_4 είναι ξεκάθαρη, όντας ικανό να ανιχνεύσει μέχρι και το 90% των αντίστοιχων κακόβουλων πακέτων την ίδια στιγμη που οι επιδόσεις του αυτοκωδικοποιητή αντιστοιχούν σε τιμές TPR χαμηλότερες του 1%.

Σχήμα 4.4: Γραφική αναπαράσταση των αποτελεσμάτων του Πίνακα 4.14, καθώς και των αντίστοιχων ποσοστών TNR.

(α′) Ποσοστά TNR




(β′) Ποσοστά TPR (Πίνακας 4.14)




Τέλος θα μπορούσαμε να αναφερθούμε συνοπτικά στην διαδικασία της εκπαίδευσης των μοντέλων RNN και AE-1. Σχετικά με την εκπαίδευση των συστημάτων HTM το μόνο που θα πρέπει να αναφέρουμε είναι το γεγονός ότι ορίζουμε μία και μόνο εποχή εκπαίδευσης, καθώς όπως θα δούμε αργότερα στο πέμπτο κεφάλαιο της εργασίας τα συστήματα HTM πάσχουν από ορισμένους χρονικούς περιορισμούς, οι οποίοι επιδεινώνονται ακόμη περισσότερο στην περίπτωση που έχει προηγηθεί μία διαδικασία εκπαίδευσης μακράς διαρκείας. Κλείνοντας αυτήν την μικρή παρένθεση, στον Πίνακα 4.15 παρουσιάζουμε τις παραμέτρους εκπαίδευσης των δύο μοντέλων των νευρωνικών δικτύων.

Πίνακας 4.15: Οι τιμές των παραμέτρων εκπαίδευσης των μοντέλων RNN-1 και AE-1 στα πλαίσια χρήσης των συνόλων δεδομένων των πακέτων.

| Παράμετροι / Μοντέλο | **Optimizer** | **Loss Fun.** | **Εποχές** | **Ρυθμός Μάθ.** | **Batch Size** | **L2 Weight** |
|---|---|---|---|---|---|---|
| **RNN-1** | Adam | Cross Entropy | 250 | $5 \cdot 10^{-4}$ | 128 | $1 \cdot 10^{-3}$ |
| **AE-1** | Adam | MSE | 250 | $1 \cdot 10^{-3}$ | 128 | $1 \cdot 10^{-1}$ |

Γενικότερα θα λέγαμε ότι η εκπαίδευση των δικτύων δεν αποτέλεσε επίπονη διαδικασία καθώς και τα δύο μοντέλα με ευκολία επέτευξαν υψηλές επιδόσεις της τάξης του 99%, τουλάχιστον αναφορικά με τις επιθέσεις των Booter_1, 2, 3, 6 και 7. Το φαινόμενο αυτό βέβαια ενδέχεται να οφείλεται σε μεγάλο βαθμό στο γεγονός

ότι το πλήθος των μοναδικών στοιχείων των συνόλων είναι αρκετά μικρό, όπως εξάλλου φαίνεται και από τον Πίνακα 4.5. Για αυτόν τον λόγο δεν χρειάστηκε να χρησιμοποιήσουμε κάποια πιο σύνθετη μέθοδο επιλογής βέλτιστου μοντέλου, σε αντίθεση με την περίπτωση των δεδομένων των ροών πακέτων, όπως θα δούμε παρακάτω.

### 4.5.2 Τα Αποτελέσματα των Πειραμάτων βάσει των Δεδομένων των Ροών Πακέτων

Ξεκινάμε και πάλι την παρουσιαση με την περίπτωση της Επιβλεπόμενης Μάθησης, στα πλαίσια της οποίας υπενθυμίζουμε ότι τα μοντέλα RNN-1 και HTM εκπαιδεύονται βάσει ενός μικτού συνόλου εκπαίδευσης αποτελούμενο από 173,368 στοιχεία, τα οποία σχετίζονται με έξι διαφορετικές επιθέσεις. Στην συνέχεια κάθε μοντέλο αξιολογείται βάσει ενός επίσης μικτού συνόλου δεδομένων μεγέθους 5,696,426, το οποίο ωστόσο περιέχει έξι άγνωστες επιπλέον επιθέσεις.

Συγκρίνοντας αρχικά τα ποσοστά TNR που επιτυγχάνουν τα δύο μοντέλα η διαφορά που παρατηρείται είναι αρκετά μικρή, με το σύστημα HTM να κατέχει ένα μικρό προβάδισμα καταφέρνοντας να ταξινομήσει επιτυχώς ένα ποσοστό 94.34%($\pm$0.70) των καλόβουλων στοιχείων, ενώ το αντίστοιχο ποσοστό που επιτυγχάνει το δίκτυο RNN-1 αγγίζει το 92.88%($\pm$1.57). Ωστόσο κάμποσες διαφορές παρατηρούνται αναφορικά με την ικανότητα των δύο μοντέλων να ανιχνεύσουν την κακόβουλη κυκλοφορία.

Πίνακας 4.16: Οι τιμές TPR (%) που επιτυγχάνουν τα δύο μοντέλα RNN-1 και HTM ως προς τις έξι γνωστές επιθέσεις κάνοντας χρήση των δεδομένων των ροών πακέτων στα πλαίσια της Επιβλεπόμενης Μάθησης.

| Μοντέλο \ Επίθεση | **LDAP** | **MSSQL** | **NetBIOS** | **Syn** | **UDP** | **UDPLag** |
|---|---|---|---|---|---|---|
| **RNN-1** | 99.81% (± 0.04) | 99.95% (± 0.01) | 99.98% (± 0.01) | 99.23% (± 0.40) | 99.77% (± 0.03) | 97.23% (± 1.40) |
| **HTM** | 58.80% (± 0.04) | 99.94% (± 0.00) | 0.29% (± 0.07) | 99.78% (± 0.22) | 99.31% (± 0.45) | 99.58% (± 0.20) |

Οπως φαίνεται και από τον Πίνακα 4.16, ενώ το δίκτυο RNN-1 επιτυγχάνει ποσοστά TPR άνω του 97% σε κάθε μία από τις έξι γνωστές επιθέσεις, το σύστημα HTM από την άλλη συναντά δυσκολίες όσο αφορά την ανίχνευση των επιθέσεων LDAP και NetBIOS. Για την δεύτερη συγκεκριμένα το ποσοστό επιτυχούς ταξινόμησης είναι μικρότερο του 1%. Παρόμοια αποτελέσματα παρατηρούμε και στην περίπτωση των άγνωστων επιθέσεων, με τα αντίστοιχα αποτελέσματα να παρουσιάζονται στον Πίνακα 4.17.

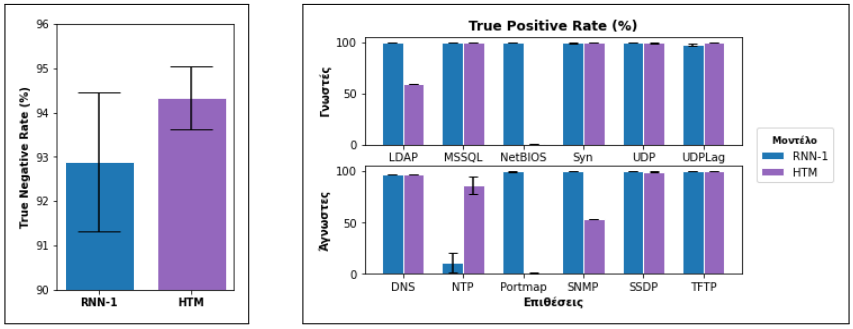
Πίνακας 4.17: Οι τιμές TPR (%) που επιτυγχάνουν τα δύο μοντέλα RNN-1 και HTM ως προς τις έξι άγνωστες επιθέσεις κάνοντας χρήση των δεδομένων των ροών πακέτων στα πλαίσια της Επιβλεπόμενης Μάθησης.

| Μοντέλο \ Επίθεση | **DNS** | **NTP** | **Portmap** | **SNMP** | **SSDP** | **TFTP** |
|---|---|---|---|---|---|---|
| **RNN-1** | 96.91% (± 0.12) | 10.79% (± 9.19) | 99.63% (± 0.04) | 99.84% (± 0.00) | 99.96% (± 0.00) | 99.87% (± 0.01) |
| **HTM** | 97.11% (± 0.27) | 86.58% (± 8.34) | 0.74% (± 0.11) | 53.57% (± 0.05) | 99.50% (± 0.31) | 99.87% (± 0.03) |

Συγκεκριμένα το σύστημα HTM φαίνεται να συναντά δυσκολίες κατά την ανίχνευση των δύο επιθέσεων Portmap και SNMP, ενώ το δίκτυο RNN-1 συνεχίζει να διατηρεί τα προηγούμενα υψηλά ποσοστά TPR, τουλάχιστον ως προς τις πέντε από τις έξι συνολικά άγνωστες επιθέσεις, καθώς στα πλαίσια της ανίχνευσης της επίθεσης NTP ξεκάθαρα παρατηρούμε την υπεροχή του συστήματος HTM. Ωστόσο αξίζει να σημειώσουμε ότι η επίθεση αυτή φαίνεται να προκαλεί την μεγαλύτερη σύγχυση στα μοντέλα, δεδομένου ότι μαζί της συνδέονται οι δύο υψηλότερες τιμές τυπικής απόκλισης των τιμών TPR, 8.34% για το σύστημα HTM και 9.19% για το μοντέλο RNN-1. Τα αποτελέσματα αυτά παρουσιάζονται γραφικά μέσω του Σχήματος 4.5. Γενικότερα θα λέγαμε πως και τα δύο μοντέλα έχουν κάνει μία αρκετά καλή δουλειά όσο αφορά την ανίχνευση της κακόβουλης κίνησης, ιδιαίτερα εάν λάβουμε υπόψη ότι στην περίπτωση των έξι αγνώστων επιθέσεων τα αντίστοιχα δεδομένα δεν συμμετείχαν στο σύνολο εκπαίδευσης.

Σχήμα 4.5: Γραφική αναπαράσταση των αποτελεσμάτων των Πινάκων 4.16 και 4.17, καθώς και των αντίστοιχων ποσοστών TNR.

(α′) Ποσοστά TNR (β′) Ποσοστά TPR (Πίνακες 4.16 και 4.17)




Στην συνέχεια θα εξετάσουμε τα αποτελέσματα που εξήχθησαν μέσω της εκτέλεσης των πειραμάτων Μη-Επιβλεπόμενης Μάθησης, στα πλαίσια των οποίων τα μοντέλα AE-1 και HTM εκπαιδεύτηκαν βάσει μόλις 11,547 καλόβουλων στοιχείων. Ξεκινώντας από τις τιμές TNR, αυτήν τη φορά νικητής αναδεικνύεται το δίκτυο AE-1 με ποσοστό 93.21%(±0.38), ενώ το σύστημα HTM –με αντίστοιχο ποσοστό 80.94%(±0.23)– φαίνεται να βρίσκεται αρκετά πίσω, ταξινομώντας λανθασμένα ένα ποσοστό της τάξης του 20% των στοιχείων που αποτελούν μέρος της καλόβουλης κίνησης. Αναφορικά με τις τιμές των ποσοστών TPR των δύο μοντέλων, αυτές παρουσιάζονται λεπτομερώς στους Πίνακες 4.18 και 4.19.

Πίνακας 4.18: Οι τιμές TPR (%) που επιτυγχάνουν τα δύο μοντέλα AE-1 και HTM ως προς τις έξι γνωστές επιθέσεις κάνοντας χρήση των δεδομένων των ροών πακέτων στα πλαίσια της Μη-Επιβλεπόμενης Μάθησης.

| Μοντέλο \ Επίθεση | LDAP | MSSQL | NetBIOS | Syn | UDP | UDPLag |
|---|---|---|---|---|---|---|
| **AE-1** | 99.90% (± 0.00) | 99.98% (± 0.00) | 99.83% (± 0.00) | 0.03% (± 0.00) | 99.97% (± 0.00) | 23.20% (± 0.00) |
| **HTM** | 99.93% (± 0.02) | 99.98% (± 0.00) | 99.91% (± 0.05) | 0.03% (± 0.00) | 99.98% (± 0.00) | 23.22% (± 0.00) |

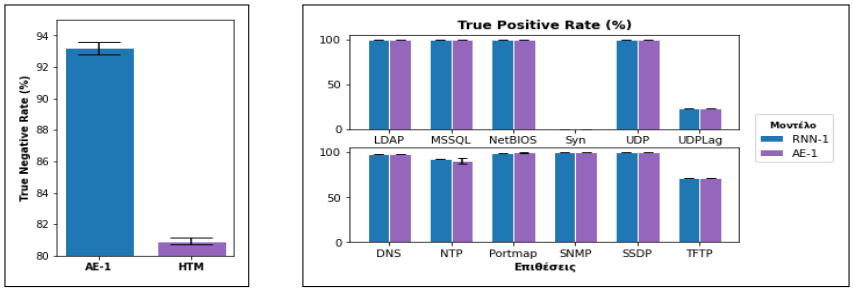
Πίνακας 4.19: Οι τιμές TPR (%) που επιτυγχάνουν τα δύο μοντέλα AE-1 και HTM ως προς τις έξι άγνωστες επιθέσεις κάνοντας χρήση των δεδομένων των ροών πακέτων στα πλαίσια της Μη-Επιβλεπόμενης Μάθησης.

| Μοντέλο \ Επίθεση | DNS | NTP | Portmap | SNMP | SSDP | TFTP |
|---|---|---|---|---|---|---|
| **AE-1** | 98.10% (± 0.05) | 93.03% (± 0.00) | 99.50% (± 0.00) | 99.80% (± 0.00) | 99.91% (± 0.00) | 71.80% (± 0.00) |
| **HTM** | 98.20% (± 0.18) | 90.73% (± 3.28) | 99.60% (± 0.05) | 99.86% (± 0.03) | 99.92% (± 0.00) | 71.81% (± 0.00) |

Αρχικά θα πρέπει να τονίσουμε ότι η ομαδοποίηση των επιθέσεων στους δύο Πίνακες 4.18 και 4.19 βάσει του ίδιου μοτίβου των «γνωστών» και «άγνωστων» επιθέσεων πραγματοποιείται αποκλειστικά και μόνο για λόγους ευμορφίας, εφόσον στα πλαίσια των πειραμάτων Μη-Επιβλεπόμενης Μάθησης κάθε επίθεση θεωρείται ούτως ή άλλως άγνωστη. Σχετικά τώρα με τον σχολιασμό των αποτελεσμάτων θα λέγαμε ότι τα ποσοστά TPR που επιτυγχάνουν τα δύο μοντέλα σχεδόν ταυτίζονται, κάτι το οποίο διακρίνεται ακόμη καλύτερα μέσω του Σχήματος 4.6 (β′). Η μόνη διαφορά που παρατηρείται αφορά την επίθεση NTP, στα πλαίσια της οποίας υπερτερεί το δίκτυο AE-1 με μία ωστόσο αμελητέα διαφορά της τάξης του 2%-3%.

Σχήμα 4.6: Γραφική αναπαράσταση των αποτελεσμάτων των Πινάκων 4.18 και 4.19, καθώς και των αντίστοιχων ποσοστών TNR.

(α′) Ποσοστά TNR (β′) Ποσοστά TPR (Πίνακες 4.18 και 4.19)

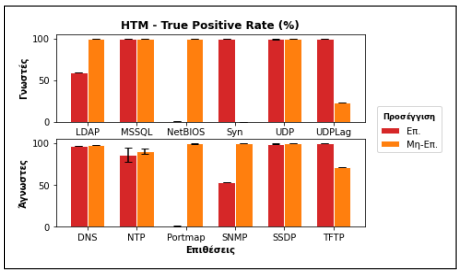



Αν εξαιρέσουμε τις επιθέσεις Syn και UDPLag για τις οποίες τα αντίστοιχα ποσοστά TPR είναι υπερβολικά χαμηλά, και τα δύο μοντέλα κατορθώνουν αρκετά υψηλές επιδόσεις δεδομένου ότι το σύνολο εκπαίδευσής τους αποτελείται από λιγότερα των 12,000 καλόβουλων στοιχείων. Εάν ωστόσο έπρεπε να επιλέξουμε μονάχα ένα μεταξύ των δύο μοντέλων, αυτό θα ήταν ο αυτοκωδικοποιητής AE-1 λόγω του υψηλότερου ποσοστού TNR που επιτυγχάνει, καθώς η λανθασμένη ταξινόμηση της καλόβουλης κίνησης ως κακόβουλη μπορεί να αποβεί εξίσου καταστροφική με την αδυναμία ανίχνευσης μίας επίθεσης.

Όσο αφορά τώρα την σύγκριση των δύο προσεγγίσεων, αυτές της Επιβλεπόμενης και της Μη-Επιβλεπόμενης Μάθησης, δεν θα λέγαμε ότι παρατηρούμε υπερβολικά μεγάλες διαφορές ως προς τα αποτελέσματά τους. Ιδιαίτερα για την περίπτωση των δύο δικτύων RNN-1 και AE-1, τόσο τα ποσοστά TNR όσο και τα ποσοστά TPR βρίσκονται αρκετά κοντά μεταξύ τους με το δίκτυο RNN-1 να παρουσιάζει ελαφρώς υψηλότερες επιδόσεις, κάτι το οποίο ωστόσο θα έπρεπε να περιμένουμε εφόσον το δίκτυο αυτό αξιοποιεί την πληροφορία των ετικετών.

Από την άλλη, συγκρίνοντας τις δύο προσεγγίσεις από την σκοπιά του συστήματος HTM δεν μπορούμε να εξάγουμε κάποιο συγκεκριμένο συμπέρασμα βάσει των αποτελεσμάτων που σχετίζονται με την ανίχνευση των επιθέσεων, καθώς δεν διακρίνεται κάποιο ιδιαίτερο μοτίβο συμπεριφοράς. Όπως φαίνεται και από το Σχήμα 4.7 στα δεξιά, ορισμένες επιθέσεις ανιχνεύονται ευκολότερα μέσω της πρώτης προσέγγισης ενώ άλλες μέσω της δεύτερης. Συγκρίνοντας ωστόσο τις δύο μεθόδους ως προς τις τιμές των ποσοστών TNR τότε σαφώς και θα επιλέγαμε την μέθοδο της Επιβλεπόμενης Μάθησης, χάρη στην οποία το σύστημα HTM ταξινομεί επιτυχώς μέχρι και 13% περισσότερες καλόβουλες ροές πακέτων σε σχέση με την αντιστοιχη περίπτωση της Μη-Επιβλεπόμενης Μάθησης.

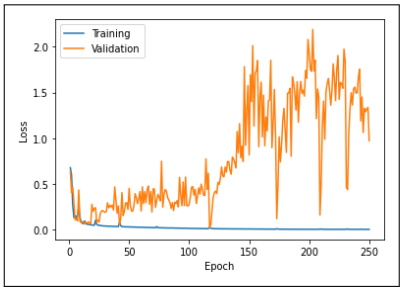
Σχήμα 4.7: Σύγκριση των τιμών TPR (%) που επιτυγχάνονται από το σύστημα HTM μέσω των προσεγγίσεων της Επιβλεπόμενης και της Μη-Επιβλεπόμενης Μάθησης.




Τέλος θα κλείσουμε αυτό το κεφάλαιο με μία αναφορά στην διαδικασία εκπαίδευσης των δικτύων RNN-1 και AE-1, η οποία αποδείχτηκε ελαφρώς πιο περίπλοκη από ότι στην αντίστοιχη περίπτωση των δεδομένων των πακέτων. Για αυτό το λόγο έχουμε κατασκευάσει δύο ξεχωριστά σύνολα επικύρωσης, ένα για κάθε μοντέλο. Το μικτό σύνολο επικύρωσης μεγέθους 25,000 στοιχείων το οποίο χρησιμοποιείται κατά την εκπαίδευση του δικτύου RNN-1 παρουσιάζεται αναλυτικότερα στον Πίνακα 4.8, ενώ το αντίστοιχο σύνολο δεδομένων για την περίπτωση της μη-επιβλεπόμενης μάθησης αποτελείται από 6,678 ροές πακέτων οι οποίες αποτελούν μέρος της συνηθισμένης καλόβουλης κίνησης.

Το μεγαλύτερο πρόβλημα συγκεκριμένα παρατηρήθηκε κατά την εκπαίδευση του μοντέλου RNN-1, κάτι το οποίο διακρίνεται και από το Σχήμα 4.8 στο οποίο απεικονίζεται η εξέλιξη των σφαλμάτων εκπαίδευσης και επικύρωσης που παράγονται μέσω του μοντέλου από εποχή σε εποχή. Όπως παρατηρούμε και από το σχήμα στα δεξιά, ενώ το σφάλμα εκπαίδευσης ακολουθεί μία φυσιολογική πορεία παρουσιάζοντας μία απότομη μείωση κατά το αρχικό στάδιο της εκπαίδευσης η οποία στην συνέχεια σταδιακά εξασθενεί, η πορεία του σφάλματος επικύρωσης από την άλλη μοιάζει χαοτική. Συγκεκριμένα η τιμή αυτή μειώνεται περίπου μέχρι και την εικοστή εποχή, ενώ στην συνέχεια ακολουθεί μία άκρως ασταθή αυξητική πορεία, κάτι το οποίο σαφώς δεν

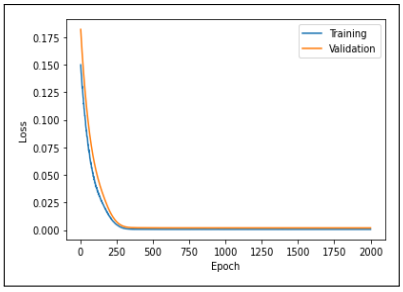
Σχήμα 4.8: Το Γράφημα του σφάλματος εκπαίδευσης/επικύρωσης του μοντέλου RNN-1 ως προς το πλήθος των εποχών εκπαίδευσης βάσει των δεδομένων των ροών πακέτων.



επιθυμούμε. Για αυτόν τον λόγο το τελικό μοντέλο επιλέγεται ως εξής: Στο τέλος κάθε εποχής υπολογίζουμε την ακρίβεια[34] του μοντέλου χρησιμοποιώντας το σύνολο επικύρωσης, την οποία και αποθηκεύουμε μαζί με τους πίνακες κατάστασης των βαρών του δικτύου την εκάστοτε στιγμή. Μετά την ολοκλήρωση της εκπαίδευσης ανακτούμε τα βάρη του δικτύου μέσω των οποίων επιτεύχθηκε η μέγιστη ακριβεια, ούτως ώστε να τα εφαρμόσουμε στο τελικό μοντέλο. Μέσω αυτής της διαδικασίας εγγυόμαστε ότι τελικό μοντέλο θα είναι όσο το δυνατόν πιο ικανό να γενίκευσει την γνώση του πάνω σε νέα δεδομένα.

Αναφορικά με την περίπτωση της Μη-Επιβλεπόμενης Μάθησης, μολονότι η εκπαίδευση του μοντέλου AE-1 αποδείχτηκε αρκετά ομαλότερη, όπως φαίνεται εξάλλου και από το Σχήμα 4.9, χρησιμοποιούμε ωστόσο την ίδια μέθοδο επιλογής του μοντέλου που περιγράφουμε παραπάνω. Παρόλα αυτά στην περίπτωση του αυτοκωδικοποιητή συναντάμε ένα εμπόδιο το οποίο συνίσταται στον υπολογισμό της τιμής του κατωφλιού ανωμαλίας, καθώς για να υπολογίσουμε την ακρίβεια του μοντέλου στο τέλος κάθε εποχής θα πρέπει πρώτα να επαναϋπολογίσουμε την εκάστοτε τιμή του κατωφλιού βάσει ολόκληρου του συνόλου εκπαίδευσης. Αν και στην περίπτωσή μας το σύνολο εκπαίδευσης είναι αρκετά μικρό, παράλληλα χρησιμοποιούμε ένα σχετικά μεγάλο πλήθος εποχών καθώς η διαδικασία εκπαίδευσης έδειξε να εξελίσσεται με αργότερους ρυθμούς. Στην θέση της ακρίβειας επομένως χρησιμοποιούμε το σφάλμα επικύρωσης του μοντέλου –το οποίο είναι σαφώς ευκολότερο να υπολογίσουμε–, ενώ μετά την ολοκλήρωση της εκπαίδευσης διατηρούμε τα βάρη εκείνα τα οποία αντιστοιχούν στην επίτευξη της ελάχιστης τιμής του εν λόγω σφάλματος.

Σχήμα 4.9: Το Γράφημα του σφάλματος εκπαίδευσης/επικύρωσης του μοντέλου AE-1 ως προς το πλήθος των εποχών εκπαίδευσης βάσει των δεδομένων των ροών πακέτων.




Όπως φάνηκε και από τα παραπάνω αποτελέσματα, η μέθοδος αυτή της επιλογής του μοντέλου λειτουργεί αρκετά καλά καθώς τα μοντέλα των νευρωνικών δικτύων θα λέγαμε ότι σε γενικές γραμμές παρουσίασαν καλύτερες επιδόσεις από ότι τα συστήματα HTM. Τέλος, στον Πίνακα 4.20 παρουσιάζουμε τις τιμές των παραμέτρων εκπαίδευσης καθενός από τα δύο δίκτυα λεπτομερώς.

Πίνακας 4.20: Οι τιμές των παραμέτρων εκπαίδευσης των μοντέλων RNN-1 και AE-1 στα πλαίσια χρήσης των συνόλων δεδομένων των ροών πακέτων.

| Παράμετροι / Μοντέλο | **Optimizer** | **Loss Fun.** | **Εποχές** | **Ρυθμός Μάθ.** | **Batch Size** | **L2 Weight** |
|---|---|---|---|---|---|---|
| **RNN-1** | Adam | Cross Entropy | 250 | $5 \cdot 10^{-4}$ | 32 | $1 \cdot 10^{-5}$ |
| **AE-1** | Adam | MSE | 2000 | $1 \cdot 10^{-4}$ | 16 | $5 \cdot 10^{-4}$ |

[34] Η μετρική της ακρίβειας υπολογίζεται ως $\frac{\text{TP+TN}}{\text{TP+FP+TN+FN}}$.

# Κεφάλαιο 5

# Συμπεράσματα & Επεκτάσεις

Σε αυτήν την εργασία εφαρμόσαμε δύο διαφορετικά είδη μοντέλων Μηχανικής Μάθησης, τα νευρωνικά δίκτυα και τα συστήματα HTM, με σκοπό την ανίχνευση διαφορετικών ειδών διαδικτυακών επιθέσεων DDoS εστιάζοντας παράλληλα στην κατανόηση της λειτουργίας των συστημάτων HTM αλλά και στους διάφορους τρόπους εφαρμογής αυτών των μοντέλων ως προς την επίλυση προβλημάτων Μηχανικής Μάθησης, επιβλεπόμενης και μη. Στο τελευταίο κεφάλαιο της εργασίας θα επιχειρήσουμε να συνοψίσουμε τα συμπεράσματα που έχουν εξαχθεί κατά την εκπόνηση αυτής της εργασίας, τόσο μέσω της θεωρητικής μελέτης των συστημάτων HTM όσο και πειραματικά.

Οπως έδειξαν και τα πειράματα του προηγούμενου κεφαλαίου τα συστήματα HTM είναι ικανά να ανταγωνιστούν ακόμη και τα μοντέλα των νευρωνικών δικτύων, τουλάχιστον στα πλαίσια του προβλήματος που εξετάζουμε. Αν και πρέπει να παραδεχτούμε πως σε γενικές γραμμές τα νευρωνικά δίκτυα κατόρθωσαν υψηλότερες επιδόσεις, η επίδοση των συστημάτων HTM δεν απείχε πολύ από αυτή των νευρωνικών δικτύων, ενώ υπήρξαν ακόμη και περιπτώσεις όπου τα συστήματα HTM υπερτερούσαν. Ωστόσο δεν θα πρέπει να αποφύγουμε να αναφέρουμε μερικά αρνητικά σημεία των μοντέλων αυτών, τα οποία ενδέχεται να τα καταστήσουν ακατάλληλα για χρήση σε ορισμένες περιπτώσεις.

Το πρώτο πρόβλημα που παρατηρούμε είναι ο χρόνος υπολογισμού της εξόδου. Ακόμη και στην περίπτωσή μας όπου το σύστημα HTM το οποίο κατασκευάσαμε αποτελείται από μία και μοναδική περιοχή, το χρονικό διάστημα που μεσολαβεί μεταξύ της τροφοδότησης της εισόδου στο μοντέλο και της παραγωγής της συμβολοσειράς εξόδου είναι αρκετά μεγαλύτερο από το αντίστοιχο χρονικό διάστημα παραγωγής της εξόδου μέσω ενός νευρωνικού δικτύου. Εκτός από αυτό, ο χρόνος υπολογισμού της εξόδου ενός συστήματος HTM συνεχίζει να αυξάνεται καθώς το σύστημα αυτό εκπαιδεύεται. Το γεγονός αυτό οφείλεται στην δημιουργία νέων περιφερικών τμημάτων δενδρίτη καθώς και νέων συνάψεων κατά την διάρκεια της

εκπαίδευσης, το αυξημένο μέγεθος των οποίων επιβαρύνει χρονικά την εκτέλεση του αλγορίθμου Temporal Pooling.

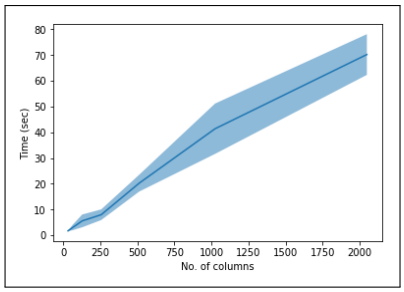
Στον Πίνακα 5.1 μπορούμε να διακρίνουμε την επίδραση αυτού του φαινομένου, καθώς το μέγεθος του συνόλου εκπαίδευσης και ο χρόνος υπολογισμού της εξόδου μοιάζουν να συσχετίζονται γραμμικά μεταξύ τους. Για παράδειγμα ένα σύστημα HTM το οποίο έχει εκπαιδευτεί βάσει ενός συνόλου δεδομένων μεγέθους 10,000 στοιχείων χρειάζεται περίπου δώδεκα δευτερόλεπτα ούτως ώστε να παράγει την έξοδο για κάθε στοιχείο ενός συνόλου δεδομένων ίδιου μεγέθους. Αντιθέτως, στην περίπτωση που το ίδιο σύστημα HTM έχει εκπαιδευτεί βάσει 250,000 στοιχείων, ο αντίστοιχος χρόνος υπολογισμού αυξάνεται από τα δώδεκα δευτερόλεπτα στα εκατό δεκαέξι, ή αλλιώς στα δύο περίπου λεπτά.

Πίνακας 5.1: Η επιρροή του μεγέθους του συνόλου εκπαίδευσης ενός συστήματος HTM πάνω στον χρόνο υπολογισμού της εξόδου σε δευτερόλεπτα.

| Μέγεθος Συνόλου Υπολογισμού Εξόδου / Μέγεθος Συνόλου Εκπαίδευσης | **10k** | **25k** | **50k** | **100k** | **250k** |
|---|---|---|---|---|---|
| **10k** | 12.46 | 31.18 | 63.01 | 127.00 | 324.06 |
| **25k** | 16.55 | 41.46 | 81.99 | 164.14 | 418.60 |
| **50k** | 32.78 | 72.45 | 168.48 | 336.71 | 888.83 |
| **100k** | 56.57 | 143.21 | 284.33 | 559.41 | 1,287.01 |
| **250k** | 116.21 | 296.34 | 601.42 | 1,224.87 | 3,015.81 |

Όσο αφορά το μήκος των συμβολοσειρών εισόδου, η τιμή αυτής της παραμέτρου δεν φάνηκε να έχει κάποια σημαντική επίδραση πάνω στον υπολογιστικό χρόνο των μοντέλων. Εάν έπρεπε ωστόσο να επιλέξουμε κάποια παράμετρο η τιμή της οποίας επηρεάζει σε μεγάλο βαθμό τον χρόνο αυτό, τότε η παράμετρος αυτή θα ήταν αναμφίβολα το πλήθος των στηλών των περιοχών του συστήματος. Όπως παρατηρούμε και στο Σχήμα 5.1, ο χρόνος υπολογισμού της εξόδου φαίνεται να συνδέεται γραμμικά με το πλήθος των στηλών του συστήματος. Στο συγκεκριμένο παράδειγμα βέβαια το σύστημα HTM αποτελείται από μία και μόνο περιοχή, συνεπώς όλες οι στήλες υπάγονται σε αυτή. Σχετικά με την σχέση του χρόνου υπολογισμού της εξόδου και του πλήθους των περιοχών είναι ξεκάθαρο πως αυτή είναι επίσης γραμμική, καθώς η ύπαρξη $n$ περιοχών συνεπάγεται την εκτέλεση των ίδιων βημάτων $n$ φορές. Ωστόσο όπως έχουμε ήδη αναφέρει, μία περιοχή είναι αρκετή για την αντιμετώπιση των περισσότερων προβλημάτων.

Σχήμα 5.1: Το Γράφημα του χρόνου υπολογισμού της εξόδου βάσει 10,000 στοιχείων ως προς το πλήθος των στηλών ενός συστήματος HTM[1] μοναδικής περιοχής.




[1] Στην πραγματικότητα ο χρόνος υπολογισμού υπολογίζεται ως ο μέσος χρόνος υπολογισμού βάσει τριών συστημάτων HTM ίδιας αρχιτεκτονικής. Η αντίστοιχη τυπική απόκλιση παρουσιάζεται μέσω της γαλάζιας περιοχής γύρω από την μπλε γραμμή.

Τα παραπάνω αποτελέσματα σαφώς υποδεικνύουν την μειονεκτική θέση που κατέχουν τα συστήματα HTM όσο αφορά τον υπολογιστικό χρόνο έναντι των νευρωνικών δικτύων, τα οποία είναι ικανά να υπολογίσουν την έξοδο χιλιάδων στοιχείων σε κλάσματα δευτερολέπτων. Χάρη στην ίδια την φύση των υπολογιστικών πράξεων που εκτελούν τα νευρωνικά δίκτυα, το μεγαλύτερο μέρος των οποίων αφορά πράξεις πολλαπλασιασμού πινάκων, τα μοντέλα αυτά μπορούν να εκμεταλλευτούν κατάλληλα διαμορφωμένο υλικό (hardware), όπως για παράδειγμα αποτελούν οι κάρτες γραφικών (GPU), ούτως ώστε να μειώσουν τον υπολογιστικό τους χρόνο ακόμη περισσότερο. Αντιθέτως στην περίπτωση των συστημάτων HTM η διαδικασία αυτή μοιάζει δυσκολότερη καθώς η υλοποίησή τους εξαρτάται σε έναν αρκετά μεγάλο βαθμό από την εκτέλεση επαναληπτικών βρόχων και λογικών πράξεων. Θα πρέπει φυσικά να αναφέρουμε ότι πραγματοποιούνται κάμποσες προσπάθειες από την ευρύτερη κοινότητα των συστημάτων HTM για την παραλληλοποίησή ορισμένων από τους υπολογισμούς, οι οποίες ωστόσο απέχουν αρκετά από το να ολοκληρωθούν στο άμεσο μέλλον. Συνεπώς καταλήγουμε ότι η χρήση αυτών των μοντέλων θα πρέπει να αποφεύγεται σε περιπτώσεις όπου η ταχύτατη επεξεργασία των δεδομένων είναι υψίστης σημασίας.

Ένα δεύτερο πρόβλημα το οποίο προκύπτει κατά την εφαρμογή των συστημάτων HTM είναι η προσαρμογή των τιμών των παραμέτρων. Ακόμη και σε ένα μοντέλο το οποίο απότελείται από μία και μοναδική περιοχή αντιστοιχούν είκοσι πέντε διαφορετικές παράμετροι, εκ των οποίων οι δεκατρείς σχετίζονται άμεσα με τον αλγόριθμο Spatial Pooler, οι δέκα αντίστοιχα με τον αλγόριθμο Temporal Pooler, ενώ οι υπόλοιπες δύο αφορούν την γενικότερη αρχιτεκτονική της περιοχής του συστήματος, συγκεκριμένα το πλήθος των στηλών αλλά και πλήθος των κυττάρων ανά στήλη. Εκτός όμως από τις παραπάνω παραμέτρους δεν θα πρέπει να ξεχάσουμε το γεγονός πως κάθε χαρακτηριστικό της εισόδου χρήζει κωδικοποίησης, η οποία με την σειρά της επίσης εξαρτάται από ένα πλήθος παραμέτρων, όπως για παράδειγμα αποτελούν το συνολικό μήκος της συμβολοσειράς, το διάστημα κωδικοποίησης κ.λπ. Για περιπτώσεις προβλημάτων όπως το δικό μας, στα πλαίσια του οποίου καταλήξαμε σε λιγότερα από δεκαπέντε διαφορετικά χαρακτηριστικά, θα μπορούσαμε να ισχυριστούμε ότι η κωδικοποίηση των δεδομένων δεν αποτελεί ένα ιδιαίτερα σημαντικό πρόβλημα. Όταν όμως κάθε στοιχείο περιγράφεται από εκατοντάδες ή ακόμη και από χιλιάδες διαφορετικά χαρακτηριστικά, τότε εύκολα διαπιστώνουμε πως η διαδικασία της κωδικοποίησής τους παύει να αποτελεί μία τόσο απλή υπόθεση.

Λόγω αυτού του συνδυασμού της ύπαρξης ενός μεγάλου πλήθους παραμέτρων και του αυξημένου χρόνου υπολογισμού της εξόδου γίνεται λοιπόν προφανές ότι η επιλογή ενός τελικού μοντέλου το οποίο είναι ικανό να κατορθώσει όσο το δυνατόν υψηλότερος επιδόσεις ενδέχεται να αποτελέσει μία ιδιαίτερα επίπονη καθώς και χρονοβόρα διαδικασία. Ωστόσο τα παραπάνω προβλήματα δεν θα πρέπει να αποθαρρύνουν τον αναγνώστη από την χρήση των συστημάτων HTM, καθώς στην περίπτωση που η διάθεση επαρκούς χρόνου και υπολογιστικών πόρων είναι εφικτή τα προβλήματα αυτά μπορούν να ξεπεραστούν.

Η αλήθεια είναι βέβαια πως, τουλάχιστον προς το παρόν, τα μοντέλα των νευρωνικών δικτύων υπερτερούν έναντι των συστημάτων HTM, τόσο ως προς τον υπολογιστικό χρόνο όσο και ως προς τις γενικότερες επιδόσεις που επιτυγχάνουν πάνω σε προβλήματα, καθώς τα νευρωνικά δίκτυα έχουν εδραιωθεί αρκετά χρόνια τώρα ως τα ισχυρότερα μοντέλα Μηχανικής Μάθησης, το οποίο με την σειρά του επαυξάνει το κύρος τους οδηγώντας την επιστημονικη κοινότητα συνεχώς σε νέες έρευνες ως προς την περαιτέρω βελτίωσή τους. Παρόλα αυτά το γεγονός ότι τα συστήματα HTM είναι ικανά και μόνο να ανταγωνιστούν τα νευρωνικά δίκτυα, σε ορισμένες περιπτώσεις όπως είδαμε ακόμη και να τα υπερνικήσουν, αποτελεί ισχυρή ένδειξη ότι τα μοντέλα αυτά θα πρέπει επίσης να θεωρούνται υποψήφια για την αντιμετώπιση οποιουδήποτε προβλήματος. Όντας άλλωστε ένα σχετικά νέο και καινοτόμο είδος μοντέλου Μηχανικής Μάθησης, τα συστήματα HTM σαφώς και επιδέχονται επίσης βελτίωσης, πιθανώς σε μεγαλύτερο βαθμό από ότι τα μοντέλα των νευρωνικών δικτύων. Για την ακρίβεια παρακάτω παρουσιάζουμε τέσσερις διαφορετικές προτάσεις επέκτασης της δουλειάς μας, με τις οποίες θα μπορούσε να καταπιαστεί ο αναγνώστης ούτως ώστε να συμβάλλει στην περαιτέρω βελτίωση και κατανόηση των συστημάτων HTM:

- Παραλληλοποίηση του κώδικα: Όπως έχουμε αναφέρει τα συστήματα HTM υποφέρουν από έναν αρκετά υψηλό χρόνο υπολογισμού της εξόδου, γεγονός το οποίο επηρεάζει τόσο την διαδικασία της εκπαίδευσης όσο και την πρακτική τους εφαρμογή. Ο αναγνώστης επομένως θα μπορούσε να εμβαθύνει στην υπό του προγραμματιστικού πρίσματος κατανόηση των εσωτερικών μηχανισμών των συστημάτων HTM, με σκοπό την παραλληλοποίηση των υπολογισμών που πραγματοποιούνται κατά την εκτέλεση των αλγορίθμων Spatial και Temporal Pooler, μειώνοντας κατά αυτόν τον τρόπο τον χρόνο εκτέλεσής τους. Κάτι τέτοιο σαφώς είναι εφικτό καθώς η βιβλιοθήκη NuPIC αποτελεί βιβλιοθήκη ανοιχτού κώδικα.

- Ερμηνευσιμότητα των αποτελεσμάτων: Τα συστήματα HTM μπορούν στην ουσία να ιδωθούν ως μία συνάρτηση αντιστοίχησης των bit της εισόδου πάνω στα κύτταρα της τελευταίας περιοχής του συστήματος, ενσωματώνοντας σε αυτήν την αντιστοίχηση την οποιαδήποτε χωρική αλλά και χρονική πληροφορία συναντάται μεταξύ των δεδομένων. Διαφορετικά bit εισόδου –δηλαδή διαφορετικές τιμές των χαρακτηριστικών του αρχικού διανύσματος εισόδου– σε διαφορετικά χρονικά πλαίσια εντός της εκάστοτε ακολουθίας θα οδηγήσουν στην ενεργοποίηση διαφορετικών κυττάρων. Επομένως γίνεται εύκολα αντιληπτό πως επιχειρώντας να χαρτογραφήσουμε την σχέση που αναπτύσσεται μεταξύ των κυττάρων και των bit εισόδου, είναι εξαιρετικά πιθανό να μπορέσουμε να ερμηνεύσουμε τις αποφάσεις των συστημάτων HTM, όπως για παράδειγμα συμβαίνει και με απλούστερα μοντέλα Μηχανικής Μάθησης –βλέπε Δέντρα Αποφάσεων. Ωστόσο αυτή η διαδικασία σαφώς και χρήζει μεγάλης προσπάθειας και εκτεταμένης έρευνας, με την οποία θα μπορούσε να ασχοληθεί ο αναγνώστης.

- Δοκιμή περαιτέρω τεχνικών ως προς την εφαρμογή των συστημάτων HTM σε προβλήματα Μηχανικής Μάθησης: Σε αυτήν την εργασία εφαρμόσαμε τα

συστήματα HTM τόσο στα πλαίσια της Επιβλεπόμενης Μάθησης, μέσω ενός αλγορίθμου ο οποίος βασίζεται στον ταξινομητή CLA, όσο και στα πλαίσια της Μη-Επιβλεπόμενης Μάθησης, στηριζόμενοι στον μηχανισμό Έκρηξης των στηλών. Ωστόσο οι παραπάνω δύο μέθοδοι δεν αποτελούν μονόδρομο ως προς την επίλυση των δύο ειδών προβλημάτων Μηχανικής Μάθησης. Υπό αυτό το σκεπτικό ο αναγνώστης προτρέπεται να αναζητήσει εντός της σχετικής βιβλιογραφίας περισσότερες μεθόδους εφαρμογής των συστημάτων HTM, κάτι το οποίο ενδέχεται να οδηγήσει στην επίτευξη ακόμη καλύτερων αποτελεσμάτων.

- Εφαρμογή των συστημάτων HTM σε προβλήματα πρόβλεψης: Όπως είδαμε τα συστήματα HTM μέσω των κυττάρων τους έχουν την δυνατότητα να εκτελούν μελλοντικές προβλέψεις, κάτι το οποίο αν και χρήσιμο δεν αξιοποιείται στα πλαίσια αυτής της εργασίας. Ο αναγνώστης θα μπορούσε επομένως να επιχειρήσει την εφαρμογή τους ως προς την επίλυση προβλημάτων που αφορούν την πρόβλεψη μελλοντικών τιμών, για παράδειγμα του προβλήματος της πρόβλεψης χρονοσειρών.

- Διερεύνηση της βιβλιοθήκης NuPIC: Στην εργασία αυτή χρησιμοποιούμε αποκλειστικά και μόνο τέσσερις από όλες τις πιθανές κλάσεις που μας παρέχονται μέσω της βιβλιοθήκης NuPIC. Συγκεκριμένα αυτές είναι οι *ScalarEncoder* και *CategoricalEncoder* ως προς την κωδικοποίηση των τιμών των στοιχείων εισόδου, καθώς και οι κλάσεις *SpatialPooler* και *TemporalMemory* ως προς την κατασκευή πλήρως λειτουργικών περιοχών και κατ΄ επέκταση συστημάτων HTM. Ωστόσο η βιβλιοθήκη NuPIC σαφώς και περιέχει ένα μεγαλύτερο εύρος αλγορίθμων και κλάσεων, τα οποία ο αναγνώστης καλείται να διερευνήσει με σκοπό την κατασκευή ισχυρότερων συστημάτων.

Παρατηρούμε λοιπόν πως υπάρχουν αρκετές πτυχές των συστημάτων HTM οι οποίες παραμένουν ανεξερεύνητες στα πλαίσια αυτής της εργασίας, βασικός σκοπός της οποίας είναι εξάλλου να βοηθήσει τον αναγνώστη να κατανοήσει τα θεμέλια της λειτουργίας των συστημάτων αυτών, καθώς και να εισάγει ορισμένες μεθόδους βάσει των οποίων τα μοντέλα αυτά μπορούν να εφαρμοστούν ως προς την αντιμετώπιση πραγματικών προβλημάτων υπό την μορφή κλασικών προβλημάτων Μηχανικής Μάθησης, όπως για παράδειγμα στην περίπτωσή μας αποτελεί το πρόβλημα της ανίχνευσης διαδικτυακών επιθέσεων άρνησης εξυπηρέτησης. Θα πρέπει ωστόσο έως τώρα να έχει γίνει αντιληπτό πως τα οφέλη που μπορούμε να αποκομίσουμε από τα μοντέλα αυτά εκτείνονται πέραν των όσων παρουσιάζονται εντός αυτής της εργασίας. Με την σειρά μας επομένως παροτρύνουμε τον αναγνώστη να επεκτείνει αυτήν την δουλειά, είτε μέσω της ενασχόλησής του με μία ή και περισσότερες από τις τέσσερις παραπάνω προτάσεις, είτε μέσω της γενικότερης διερεύνησης των συστημάτων HTM αλλά και της αξιοποίησής τους ως μοντέλα Μηχανικής Μάθησης.

# Βιβλιογραφία


[1] Rosenblatt, F. *The Perceptron — A Perceiving and Recognizing Automaton.* Tech. Rep. 85-460-1 (Cornell Aeronautical Laboratory, 1957).

[2] Rumelhart, D., Hinton, G. & Williams, R. *Learning representations by back-propagating errors.* Nature 323, 533–536 (1986).

[3] Y. Bengio, P. Simard and P. Frasconi, *Learning long-term dependencies with gradient descent is difficult.* in IEEE Transactions on Neural Networks, vol. 5, no. 2, pp. 157-166, March 1994, doi: 10.1109/72.279181.

[4] Sepp Hochreiter and Jürgen Schmidhuber. *Long Short-Term Memory.* Neural Computation, volume 9, 1997.

[5] Dana H. Ballard (1987). *Modular learning in neural networks.* AAAI'87: Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1.

[6] Pascal Vincent, Hugo Larochelle, Yoshua Bengio & Pierre-Antoine Manzagol (2008). *Extracting and composing robust features with denoising autoencoders.* ICML '08: Proceedings of the 25th international conference on Machine learning.

[7] Diederik P. Kingma & Max Welling (2013). *Auto-Encoding Variational Bayes.* arXiv, arXiv:1312.6114.

[8] Andrew Y. Ng (2004). *Feature selection, $L_1$ vs. $L_2$ regularization, and rotational invariance.* ICML '04: Proceedings of the Twenty-First International Conference on Machine Learning.

[9] Srivastava Nitish, Hinton Geoffrey, Krizhevsky Alex, Sutskever Ilya & Salakhutdinov Ruslan. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting. The Journal of Machine Learning Research, January 2014*

[10] Hawkins, Jeff (2004). *On Intelligence* (1st ed.). Times Books. pp. 272. ISBN 978-0805074567.

[11] Lui JH, Hansen DV, Kriegstein AR (July 2011). *Development and evolution of the human neocortex*. Cell. 146 (1): 18–36. doi:10.1016/j.cell.2011.06.030. PMC 3610574. PMID 21729779.

[12] Cu Y., Ahmad S., & Hawkins J. (2017). *The HTM Spatial Pooler – a neocortical algorithm for online sparse distributed coding*. Front. Comput. Neurosci., 11, 111.

[13] Ahmad S. & Hawkins J. (2016). *How do neurons operate on sparse distributed representations? A mathematical theory of sparsity, neurons and active dendrites*. arXiv, arXiv:1601.00720.

[14] Purdy Scott (2016). *Encoding Data for HTM Systems*. arXiv, arXiv:1602.05925.

[15] Doshi, Rohan and Apthorpe, Noah and Feamster, Nick. *Machine Learning DDoS Detection for Consumer Internet of Things Devices*. 2018 IEEE Security and Privacy Workshops (SPW). http://dx.doi.org/10.1109/SPW.2018.00013

[16] M Arshi, MD Nasreen and Karanam Madhavi. *A Survey of DDOS Attacks Using Machine Learning Techniques*. E3S Web Conf., 184 (2020) 01052 DOI: https://doi.org/10.1051/e3sconf/202018401052

[17] Bindra, Naveen & Sood, Manu. (2019). *Detecting DDoS Attacks Using Machine Learning Techniques and Contemporary Intrusion Detection Dataset. Automatic Control and Computer Sciences*. 53. 419-428. 10.3103/S0146411619050043.

[18] *DDoS Attacks Increase by 151% in First Half of 2020*. Available: https://www.businesswire.com/news/home/20200916005132/en/DDoS-Attacks-Increase-by-151-in-First-Half-of-2020.

[19] H. S. Lallie, L. A. Shepherd, J. R. Nurse, A. Erola, G. Epiphaniou, C. Maple, and X. Bellekens, *Cyber security in the age of covid-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic*, arXiv preprint arXiv:2006.11929, 2020.

[20] Elsayed, Mahmoud & Le-Khac, Nhien-An & Dev, Soumyabrata & Jurcut, Anca. (2020). *DDoSNet: A Deep-Learning Model for Detecting Network Attacks*. 10.1109/WoWMoM49955.2020.00072.

[21] Salahuddin, Mohammad & Bari, Md. Faizul & Alameddine, Hyame & Pourahmadi, Vahid & Boutaba, Raouf. (2020). *Time-based Anomaly Detection using Autoencoder*. 10.23919/CNSM50824.2020.9269112.

[22] Faisal Hussain, Syed Ghazanfar Abbas, Muhammad Husnai, Ubaid Ullah Fayyaz, Farrukh Shahzad and Ghalib A. Shah. (2020). *IoT DoS and DDoS Attack Detection using ResNet*. arXiv:2012.01971.

[23] Hole, Kjell. (2016). *Anomaly Detection with HTM*. 10.1007/978-3-319-30070-2_12.

[24] Tukey, John W (1977). *Exploratory Data Analysis*. Addison-Wesley.

[25] Singh, Manmeet; Singh, Maninder; Kaur, Sanmeet (2018), *10 Days DNS Network Traffic from April-May, 2016*, Mendeley Data, V1, doi: 10.17632/zh3wnddzxy.1.

[26] Santanna, Jair & Rijswijk-Deij, Roland & Hofstede, Rick & Sperotto, Anna & Wierbosch, Mark & Granville, Lisandro & Pras, Aiko. (2015). *Booters - An analysis of DDoS-as-a-service attacks*. 243-251. 10.1109/INM.2015.7140298.

[27] *DDoS Evaluation Dataset (CIC-DDoS2019)*. `https://www.unb.ca/cic/datasets/ddos-2019.html`

[28] Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak, and Ali A. Ghorbani, *Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy*, IEEE 53rd International Carnahan Conference on Security Technology, Chennai, India, 2019

[29] Fergal Byrne, *Real Machine Intelligence with Clortex and NuPIC*, available: `https://leanpub.com/realsmartmachines/read`.