Υψηλού επιπέδου υλοποίηση των αλγορίθμων HTM ως βάση για περαιτέρω μελέτη

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Κωνσταντίνος Σαμαράς-Τσακίρης

2019-06-05

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Στόχος της εργασίας

Σε αυτήν την εργασία παρουσιάζεται μια αλγοριθμική θεωρία της νοημοσύνης, η *Hierarchical Temporal Memory*. Βασικοί αλγόριθμοι της θεωρίας διατυπώνονται σε υψηλού επιπέδου γλώσσα με εκφραστική περιεκτικότητα, αποσκοπώντας στην καταπολέμηση της βασικής μορφής πολυπλοκότητας που απαντά σε αυτό το μοντέλο: τη *γλωσσική πολυπλοκότητα* [44], δηλαδή τη *δυσκολία στην περιγραφή*.

Με προγραμματιστική διατύπωση πιστή στη μαθηματική διατύπωση των αλγορίθμων, το έργο αυτό φιλοδοξεί να θεμελιώσει και να διευκολύνει την περαιτέρω μελέτη ενός συστήματος που βασίζεται σε βιολογικές αρχές, αλλά πέρα από το ενδεχόμενο νευροεπιστημονικό του ενδιαφέρον, προσφέρεται και για την αντιμετώπιση δύσκολων προβλημάτων τεχνητής νοημοσύνης. Η Hierarchical Temporal Memory βρίσκεται σε φάση ενεργούς έρευνας και ανάπτυξης. Καθώς οι αλγόριθμοι που την περιγράφουν εξελίσσονται, μια πλατφόρμα που επιτρέπει ταχύ πειραματισμό σε επεκτάσεις και εναλλακτικές ιδέες μπορεί να διαδραματίσει σημαντικό ρόλο στην περαιτέρω μελέτη της θεωρίας. Τελικό προϊόν είναι ένα ανοιχτό πακέτο λογισμικού σε γλώσσα Julia που υλοποιεί βασικούς αλγορίθμους της θεωρίας HTM.

Για την επίτευξη του κεντρικού σκοπού, η εργασία αυτή έθεσε δύο δευτερεύοντες στόχους:

- Την κατανόηση της θεωρίας HTM
- Την εκμάθηση της γλώσσας Julia

1.2 Θεμελιώνοντας την έννοια της νοημοσύνης

Κάθε φυσικό σύστημα καθορίζεται από τους περιορισμούς στα σύνορά του. Ομοίως ο άνθρωπος καθορίζεται από την αλληλεπίδραση με το περιβάλλον του. Στην προσπάθεια να κατανοήσουμε τον άνθρωπο, το πώς λειτουργεί, το γιατί δρα με συγκεκριμένο τρόπο, σταθμό αποτελεί η κατανόηση της συμπεριφοράς που καλούμε *νοημοσύνη*.

Η συμπεριφορά είναι παρατηρήσιμη και μας δίνει μια οπτική στην εσωτερική κατάσταση, στις κρυφές μεταβλητές ενός συστήματος, οπότε ίσως αποτελεί το σημείο όπου μπορούμε να πιάσουμε το νήμα της αναζήτησης. Είναι όμως ικανοποιητικό να χαρακτηρίσουμε τη νοημοσύνη συμπεριφορά;

Ο όρος νοημοσύνη χαίρει ευρείας ερμηνείας, ευρισκόμενος στο σταυροδρόμι πολλών επιστημονικών πεδίων, από την ψυχολογία μέχρι επιστήμη υπολογιστών [12]. Όλες μάλιστα οι πιο συγκεκριμένες ερμηνείες προσπίπτουν στο ιδιαίτερα ασαφές νόημα του όρου στην καθομιλουμένη. Ίσως μπορούμε να ξεμπλέξουμε για πρακτικούς σκοπούς αυτό το κουβάρι, παρατηρώντας ότι νοημοσύνη σίγουρα επιδεικνύουν ζωντανοί οργανισμοί ως εξελικτικό χαρακτηριστικό, ως εργαλείο στη διαρκή προσαρμογή τους στο επίσης δυναμικό τους περιβάλλον.

Νοημοσύνη ως προσαρμογή

Η νοημοσύνη λοιπόν, ως προσαρμογή, προκύπτει από τη σχέση του οργανισμού με το περιβάλλον του. Αν ο οργανισμός προσαρμόζεται ταιριάζοντας υλικά του κατασκευάσματα στο περιβάλλον του, η νοημοσύνη επεκτείνει αυτήν τη δημιουργικότητα, επιτρέποντάς του να πειραματιστεί εικονικά. [2, σελ 3]. Ας ακολουθήσουμε όμως τη σκέψη του ψυχολόγου Jean Piaget στο ζήτημα.

Προσαρμογή είναι μια δυναμική διαδικασία αλληλεπίδρασης του οργανισμού με το περιβάλλον που περιλαμβάνει 2 στάδια: *αφομοίωση* και *συμβιβασμό*. Έστω ότι ο οργανισμός μπορεί να περιγραφεί με μια σειρά εσωτερικών μεταβλητών $\{a, b, c\}$ και εξωτερικών στοιχείων του περιβάλλοντος $\{x, y, z\}$, που συνδέονται μεταξύ τους με κάποιες διαδικασίες, ορίζοντας ένα μοντέλο (schema):

$$\begin{aligned} a + x &\to b \\ b + y &\to c \\ c + z &\to a \end{aligned}$$

Η *αφομοίωση* συνίσταται στην ικανότητα του οργανισμού να ενσωματώνει τα στοιχεία του περιβάλλοντος στις εσωτερικές του καταστάσεις και να συνεχίζει αυτές τις διαδικασίες. Μια αλλαγή όμως στο περιβάλλον, έστω $x \to x'$, αποτελεί πρόκληση. Είτε ο οργανισμός δεν προσαρμόζεται, που σημαίνει ότι ο κύκλος σπάει και ο οργανισμός παύει να επιτελεί κάποια λειτουργία του, είτε προσαρμόζεται, τροποποιώντας υποχρεωτικά το μοντέλο του για να *συμβιβαστεί* με τη νέα εξωγενή πραγματικότητα ($b \to b'$):

$$\begin{aligned} a + x' &\to b' \\ b' + y &\to c \\ c + z &\to a \end{aligned}$$

Σύμφωνα με αυτήν την περιγραφή, *προσαρμογή* είναι η ισορροπία της *αφομοίωσης* ($a+x \to b$...) με το *συμβιβασμό* ($x \to x' \implies b \to b'$).

Η προηγούμενη περιγραφή ισχύει εξίσου για τη *νοημοσύνη*. Νοημοσύνη είναι αφομοίωση, στο βαθμό που συμπεριλαμβάνει όλα τα εμπειρικά δεδομένα στη δομή της. Είναι όμως και συμβιβασμός, καθώς κατά τη διαρκή αφομοίωση αποκρίνεται στην πρόκληση των περιβαλλοντικών αλλαγών με τροποποίηση του μοντέλου που περιγράφει τον κόσμο.

Η διανοητική προσαρμογή λοιπόν, όπως κάθε προσαρμογή, συνίσταται από ένα μηχανισμό αφομοίωσης και συμπληρωματικού συμβιβασμού που διατηρούνται σε διαρκή ισορροπία. Ένα μυαλό προσαρμοσμένο στην πραγματικότητα είναι αυτό που δε δέχεται πια προκλήσεις στο νοητικό του μοντέλο για τον κόσμο, που δε χρειάζεται να τροποποιήσει περαιτέρω το μοντέλο αυτό για να εξηγήσει την εξελισσόμενη πραγματικότητα [2, σελ 5-7].

Ένας πρακτικός ορισμός

Από το συμπεριφορικό ορισμό του Turing στο γνωστό «Turing test» μέχρι το μηχανιστικό ορισμό του Piaget, και με πολλές στάσεις ενδιάμεσα στο [Lenat 5] και στο [Minsky 4], η πρακτικότητα της έννοιας τίθεται υπό αμφισβήτηση.

Ένας άξονας του ορισμού είναι η σχέση της νοημοσύνης με τη λογική. Στο [46] ο Wang χωρίζει τα συλλογιστικά συστήματα σε 3 κατηγορίες:

- Αμιγώς αξιωματικά. Όλες οι λογικές προτάσεις προκύπτουν από τα αξιώματα, χαρακτηριστικό παράδειγμα η ευκλείδια γεωμετρία.
- Μερικώς αξιωματικά. Η γνώση δεν επαρκεί σε όλες τις περιπτώσεις και υπάρχει μηχανισμός προσαρμογής, όπως στα ασαφή συστήματα.
- Μη αξιωματικά. Χτίζονται με βάση την υπόθεση ότι η γνώση ή οι πόροι δεν επαρκούν για οποιοδήποτε συλλογισμό.

Προτείνει λοιπόν ότι η απαίτηση νοημοσύνης ισοδυναμεί με την απαίτηση μη αξιωματικού συλλογιστικού συστήματος, λόγω της ανεπάρκειας πληροφορίας και πόρων.

Προσθέτοντας άλλο ένα έρεισμα στη συζήτηση, η νοημοσύνη συσχετίζεται άμεσα με τη δημιουργικότητα [18]. Δημιουργικότητα είναι η ικανότητα παραγωγής καινοτόμων και χρήσιμων ιδεών, επομένως συνδέεται με την τέλεση των προαναφερθέντων συμβιβασμών.

Σε αναζήτηση ενός χρήσιμου ορισμού στα πλαίσια αυτής της εργασίας, μπορούμε να στραφούμε στον ορισμό του [Wang 7]:

> Νοημοσύνη είναι η ικανότητα ενός συστήματος επεξεργασίας πληροφορίας να προσαρμόζεται στο περιβάλλον του με ανεπαρκή γνώση και πόρους.

Αναφερόμαστε σε σύστημα επεξεργασίας πληροφορίας, για να μπορούμε να μελετήσουμε την εσωτερική του κατάσταση και αλληλεπίδραση με το περιβάλλον αφηρημένα, σε αντιδιαστολή με ένα πρόβλημα π.χ. ρομποτικής. Το σύστημα έχει μια γλώσσα εισόδου και εξόδου, με την οποία εκφράζονται τα ερεθίσματα του περιβάλλοντος και οι δράσεις του συστήματος. Το σύστημα συνήθως έχει κάποιο σκοπό για τον οποίο παράγει δράσεις σύμφωνα με τη γνώση του, και για την επεξεργασία του δαπανά περιορισμένους πόρους.

Η προσαρμογή μπορεί να ερμηνευθεί όπως προηγουμένως, ή πιο συνοπτικά ως ότι το σύστημα μαθαίνει από τις εμπειρίες του.

Ο περιορισμός της ανεπαρκούς γνώσης και πόρων σημαίνει ότι το σύστημα υπόκειται σε αυτές τις συνθήκες:

- *Περιορισμένο ως προς τους υπολογιστικούς του πόρους*
- Λειτουργεί σε *πραγματικό χρόνο*
- Καλύπτει όλο το πεδίο των εκφράσιμων στη γλώσσα του εισόδων και εξόδων (*δεν υπάρχουν άκυρες είσοδοι/έξοδοι*)

Σύμφωνα με αυτόν τον ορισμό νοημοσύνη είναι μια ισχυρή μορφή προσαρμογής.

Οι παραπάνω περιορισμοί μας οδηγούν προς την εξέταση *συστημάτων ροής (streaming)*, με *ανθεκτικότητα σε σφάλματα* και με *απόδοση που να κλιμακώνεται με τους διαθέσιμους πόρους* — υπό την προϋπόθεση της *διαρκούς προσαρμογής σε νέες συνθήκες* του περιβάλλοντος.

Η αυλαία σηκώνεται για να παρουσιαστεί το υπό μελέτη σύστημα: **Hierarchical Temporal Memory (HTM) της Numenta**.

1.3 Γιατί μελετούμε την HTM;

Ο πρακτικός ορισμός της νοημοσύνης επιβάλλει περιορισμούς στο τι σύστημα θα θεωρήσουμε ότι επιδεικνύει χαρακτηριστικά νοημοσύνης. Για παράδειγμα, τα συστήματα εμπειρογνωμόνων δεν πληρούν αρκετές από τις προδιαγραφές, ενώ πολλά συστήματα νευρωνικών δικτύων επιβλεπόμενης μάθησης που βρίσκονται τώρα σε χρήση επίσης προσπίπτουν τουλάχιστον στην προδιαγραφή της διαρκούς προσαρμοστικότητας.

1.3.1 Φυσικοί αλγόριθμοι

Ακολουθώντας τη λογική του Chazelle [44], η επιτυχία της φυσικής του 20ου αιώνα είναι σε μεγάλο βαθμό η επιτυχία της μαθηματικής έκφρασης. Με ένα μικρό σύνολο εξισώσεων μπορούμε να περιγράψουμε τι συμβαίνει στο φυσικό κόσμο. Αυτή η διαπίστωση αν μη τι άλλο σκιαγραφεί τις αρχές που διέπουν το φυσικό κόσμο: συμμετρία και κανονικότητα, η αμβροσία της συνήθους μαθηματικής διατύπωσης. Αν αυτή η παρατήρηση ήταν καθολική, τα ίδια εργαλεία θα επαρκούσαν για να περιγράψουν όλα τα επιστημονικά πεδία.

Η βιολογία διέπεται προφανώς από τους ίδιους φυσικούς νόμους και συνίσταται στην εφαρμογή τους επανειλημμένα στο βάθος των αιώνων. Παρόλο που οι αρχές είναι οι ίδιες, φαινομενολογικά η υπόθεση εργασίας είναι αντίστροφη: καθε φαινόμενο είναι ειδικό και ξεχωριστό, αλλοιώσιμο υπό κάθε μετασχηματισμό, εκτός από ορισμένες περιπτώσεις που μπορούν να κατηγοριοποιηθούν μαζί. Γιατί οι αρχές επιτρέπουν καταστάσεις διακλάδωσης στα στοιχειώδη συστήματα που περιγράφουν και διάσπαση της συμμετρίας [34]. Έτσι, ο αναγωγισμός δε συνεπάγεται εποικοδομητισμό [3]. Με μια κομψή έκφραση: *η ιστορία είναι ο μεγάλος διασπαστής της συμμετρίας* [44].

Δίχως την απλοποιητική επιρροή της συμμετρίας, αυτή η οπτική γωνία έχει να αντιμετωπίσει μια μορφή πολυπλοκότητας διαφορετική από αυτήν που συνήθως ορίζουμε στην επιστήμη υπολογιστών: *εκφραστική πολυπλοκότητα*. Είναι ωφέλιμο αντιστοίχως να χρησιμοποιήσουμε και διαφορετική γλώσσα για τη μελέτη αυτών των φαινομένων: *τους (φυσικούς) αλγορίθμους* [44]. Παραδείγματα φυσικών αλγορίθμων εξερευνούνται στα [17], [24]

Συνδέοντας τον πρακτικό ορισμό της νοημοσύνης και τη χρησιμότητα των αλγορίθμων για να περιγράψουν *εκφραστικά πολύπλοκα* φαινόμενα, μπορούμε να επιχειρήσουμε τη μελέτη της νοημοσύνης με μια αλγοριθμική της θεώρηση. Αυτό είναι το βασικό κίνητρο για το μοντέλο που μελετά αυτή η εργασία, τη *Hierarchical Temporal Memory* (HTM).

1.3.2 HTM ως μοντέλο του εγκεφαλικού νεοφλοιού

Εποπτική εγκεφαλική ανατομία

Ο ανθρώπινος εγκέφαλος χωρίζεται σε διακριτές δομές με διαφορές τόσο στο μορφολογικό, όσο και στο λειτουργικό επίπεδο. Ένας χρήσιμος τέτοιος διαχωρισμός συνίσταται από τα εξής τμήματα:

- Πρόσθιος/διάμεσος εγκέφαλος, που περιλαμβάνει τα εγκεφαλικά ημισφαίρια, τους θαλάμους, τους ιπποκάμπους κ.α.
- Παρεγκεφαλίδα, που εντοπίζεται στο πίσω μέρος του κρανίου κάτω από τα ημισφαίρια
- Εγκεφαλικό στέλεχος, που εντοπίζεται επίσης προς τα πίσω και είναι η προέκταση του νωτιαίου μυελού

Σε αυτό το σημείο αξίζει μία πολύ συνοπτική επισκόπηση των εγκεφαλικών δομών, για την καλύτερη κατανόηση της HTM.

Το εγκεφαλικό στέλεχος είναι το εξελικτικά αρχαιότερο τμήμα του εγκεφάλου και σχετίζεται με βασικές ομοιοστατικές λειτουργίες. Το μεταιχμιακό σύστημα στη βάση των ημισφαιρίων αναπτύχθηκε πριν από περίπου 250 εκατομμύρια χρόνια στα θηλαστικά και μία από τις βασικές του λειτουργίες είναι η ρύθμιση των συναισθημάτων. Αυτές οι δομές έχουν μορφολογία πυρηνική, δηλαδή τα σώματα των νευρώνων τους συγκεντρώνονται σε σφαιροειδείς δομές από τις οποίες εκτείνονται οι άξονές τους.

Η επιφάνεια των εγκεφαλικών ημισφαιρίων είναι ο εγκεφαλικός φλοιός, με το μεγαλύτερο μέρος του να αποτελεί το νεοφλοιό, όπου οι νευρώνες (τα σώματα) είναι δομημένοι σε **6 επίπεδα**, και σε μικρό μέρος τον αλλοφλοιό, που έχει 3 επίπεδα νευρώνων. Οι άξονες των νευρώνων φεύγουν από το επίπεδο του φλοιού σαν καλώδια σε πλακέτα. Ο φλοιός είναι το εξελικτικά πιο σύγχρονο τμήμα του εγκεφάλου, και ειδικά ο νεοφλοιός, που απαντά μόνο σε θηλαστικά. Εδώ εντοπίζονται λειτουργίες σχετικές με «ανώτερη συλλογιστική» και αφηρημένη σκέψη. Η παρεγκεφαλίδα επίσης έχει τη μορφή φλοιού, αλλά είναι διακριτή από τα ημισφαίρια. Σχετίζεται τουλάχιστον με τη ρύθμιση λεπτών κινήσεων. Και οι δύο αυτές δομές που οργανώνονται σε φλοιούς, αντί για πυρήνες, μοιράζονται ένα γεωμετρικό πλεονέκτημα: το μέγεθός τους μπορεί να κλιμακωθεί ευκολότερα. Αποτέλεσμα της κλιμάκωσης του μεγέθους τους είναι οι αναδιπλώσεις στην επιφάνεια του ανθρώπινου εγκεφάλου. Στον άνθρωπο ο νεοφλοιός αποτελεί περίπου τα 3/4 όλου του εγκεφάλου.

Στον τομέα της μηχανικής μάθησης εμφανίζονται 3 βασικά μοντέλα μάθησης: επιβλεπόμενη, μη επιβλεπόμενη και ενισχυτική. Υπάρχουν επιχειρήματα στη βιβλιογραφία [9] ότι 3 από τις εγκεφαλικές δομές που περιγράφηκαν εφαρμόζουν αυτά τα 3 μοντέλα αντίστοιχα: η παρεγκεφαλίδα, ο εγκεφαλικός φλοιός και τα βασικά γάγγλια.

Σημαντικά ανατομικά στοιχεία του φλοιού είναι η οργάνωση των νευρώνων (εννοώντας των σωμάτων των νευρώνων) σε επίπεδα και οι αναδρομικές συνδέσεις μεταξύ τους. Έχει επίσης παρατηρηθεί ότι η πλαστικότητα των νευρικών συνάψεων στο φλοιό ακολουθεί κανόνα Hebbian: ισχυροποιούνται όταν το προσυναπτικό ερέθισμα συσχετίζεται με μετασυναπτική δραστηριότητα και εξασθενούν αλλιώς, χτίζοντας την αιτιώδη σχέση μεταξύ προσυναπτικής και μετασυναπτικής δραστηριότητας. Διατυπώνεται έτσι η υπόθεση ότι ο φλοιός μαθαίνει με μη επιβλεπόμενο τρόπο να οργανώνει την εξωγενή και εσωτερική πραγματικότητα σε έννοιες, να δημιουργεί συμβολικές αναπαραστάσεις για μεταβλητές κατάστασης.

Με δεδομένη αυτήν τη βασική ανατομία, μπορούμε να διαπιστώσουμε τι σκοπεύει να μοντελοποιήσει η HTM.

Στόχος της HTM

Η θεωρία της Hierarchical Temporal Memory αναπτύσσεται από την ερευνητική εταιρία Numenta, που δηλώνει το διττό της στόχο ως εξής: καταρχήν, την αλγοριθμική μοντελοποίηση της λειτουργίας του ανθρώπινου εγκεφαλικού νεοφλοιού, και ως συνέπεια τη μελέτη των εφαρμογών της θεωρίας τούτης ως σύστημα τεχνητής νοημοσύνης. Επομένως το μοντέλο που μελετά αυτή η εργασία δεν έχει σχεδιαστεί κατά κύριο λόγο ως σύστημα τεχνητής νοημοσύνης, αλλά ως *θεωρία της λειτουργίας του ανθρώπινου εγκεφαλικού νεοφλοιού, περιορισμένη από βιολογικά δεδομένα*. Παρόλα αυτά, εδώ δε θα συζητηθεί η νευροεπιστημονική πιστότητα του μοντέλου, μονάχα οι βιολογικές αρχές στις οποίες βασίζεται.

Σύμφωνα με τα παραπάνω ανατομικά στοιχεία, μοντελοποιώντας το νεοφλοιό η HTM δεν αποτελεί πλήρες μοντέλο του ανθρώπινου εγκεφάλου. Δεν προσφέρεται για παράδειγμα για μελέτη των συναισθημάτων ή των βασικών ομοιοστατικών μηχανισμών, αλλά μόνο των «ανώτερων συλλογιστικών». Η επιλογή αυτή δεν είναι τυχαία. Χάρη στην εξελικτική του νεότητα και γεωμετρική επεκτασιμότητα, ο νεοφλοιός δεν έχει υποστεί τόσες εξελικτικές βελτιστοποιήσεις, όσο τα υπόλοιπα τμήματα του εγκεφάλου, και διατηρεί σε μεγάλο βαθμό κοινή μορφολογία σε όλη του την έκταση. Αυτή η παρατήρηση ενδεχομένως να καθιστά το πρόβλημα της διάκρισης των θεμελιωδών αρχών λειτουργίας του από τις εξελικτικές βελτιστοποιήσεις πολύ ευκολότερο, σε σχέση με άλλα τμήματα.

Ο δευτερεύων στόχος της HTM είναι αυτός με τον οποίο ασχολείται αυτή η εργασία. Συγκεκριμένα, η HTM μπορεί να χρησιμοποιηθεί για πρόβλεψη αιτιωδών ακολουθιών (πχ χρονοσειρών) και για αναγνώριση ανωμαλιών, ή γενικότερα για την αντιμετώπιση προβλημάτων πρόβλεψης ή κατηγοριοποίησης σε μη στάσιμες ροές δεδομένων. Έχει χρησιμοποιηθεί επιτυχώς ως τώρα (ως τεχνολογία βάσης κερδοσκοπικών επιχειρήσεων) για πρόβλεψη χρηματιστηριακών δεικτών και για έγκαιρη ανίχνευση ανωμαλιών σε κέντρα δεδομένων.

1.4 Επιλογή της γλώσσας Julia

Για την παρουσιαζόμενη υλοποίηση της HTM επιλέχθηκε η σχετικά καινούρια γλώσσα επιστημονικού προγραμματισμού Julia [33].

Προτού γίνει αυτή η επιλογή, δοκιμάστηκε η υλοποίηση της HTM σε Matlab. Όμως η Matlab αποτελεί ένα κλειστό οικοσύστημα, με λίγες προοπτικές για επαναχρησιμοποίηση και ευρύ απόηχο μιας τέτοιας δουλειάς, που εξαρχής στοχεύει στην υποβοήθηση περαιτέρω έρευνας. Η Matlab έχει μακρά ιστορία στο χώρο του επιστημονικού λογισμικού και γράφτηκε το 1984 στοχεύοντας ειδικά σε υπολογιστικούς επιστήμονες και όχι σε μηχανικούς λογισμικού. Πολλές σχεδιαστικές αποφάσεις ανακλούν αυτήν την εστίαση και δημιουργούν ένα περιβάλλον ανάπτυξης λογισμικού πιο δύσχρηστο σε σχέση με εναλλακτικές όπως η Python. Η γλώσσα διευκολύνει μεν τη χρήση γραμμικής άλγεβρας, αλλά δεν επιτρέπει συναρτησιακό προγραμματισμό, «οκνηρή αποτίμηση» (lazy evaluation), ορισμό νέων τύπων δεδομένων και πολλά ακόμα στοιχεία απαραίτητα για την επιτυχία του κεντρικού στόχου αυτής της εργασίας: την εκφραστική απλότητα. Έτσι, η πρώτη υλοποίηση σε Matlab βοήθησε στην κατανόηση των αλγορίθμων, αλλά απέτυχε στον κεντρικό της στόχο.

Η Julia είναι ανοιχτό λογισμικό που ξεκίνησε από το JuliaLab του MIT το 2012 και αναπτύσσεται δημοσίως στο Github. Μόλις τον Αύγουστο του 2018 έφθασε στην πρώτη επίσημη έκδοσή της. Οι δημιουργοί της δηλώνουν ως κίνητρο για τη δημιουργία της την αντιμετώπιση του «προβλήματος των 2 γλωσσών» στην επιστημονική υπολογιστική: μία γλώσσα υψηλού επιπέδου, εύχρηστη αλλά όχι τόσο αποδοτική όπως η Python, χρησιμοποιείται αρχικά για την κατασκευή μιας πρωτότυπης λύσης· έπειτα το λογισμικό ξαναγράφεται σε μια πιο δύσχρηστη, αλλά αποδοτική γλώσσα, ενδεχομένως κατάλληλη για HPC ή για να αξιοποιήσει παραδοσιακές υπολογιστικές συστοιχίες, όπως η C++. Αυτή η ροή εργασίας είναι σύνθετη, αργή και αναποτελεσματική, απαιτώντας διπλή προσπάθεια και, συχνά, εξειδικευμένο προσωπικό. Η Julia επιδιώκει να αποτελέσει λύση σε αυτό το πρόβλημα, συνδυάζοντας την ευχρηστία της Python και την αποδοτικότητα της C++. Αν και νέα γλώσσα, έχει ήδη χτίσει ένα πλούσιο οικοσύστημα για επιστημονικό προγραμματισμό. Συμπεριλαμβάνει προφανώς τα βασικά όπως γραμμική άλγεβρα και αραιούς πίνακες. Σε μερικά πεδία όμως, όπως η αστρονομία, οι διαφορικές εξισώσεις και τα πολύπλοκα συστήματα, προσφέρει ήδη εξίσου πλήρεις ή πληρέστερες λύσεις από πιο παραδοσιακές γλώσσες, όπως η Python.

Μια επιτυχής υλοποίηση μεγάλου επιστημονικού λογισμικού που χαίρει ευρείας διαφήμισης για τη Julia είναι η Celeste [31]. Γραμμένο σε Julia, χρησιμοποιεί παραλληλισμό κοινής και κατανεμημένης μνήμης και μπόρεσε να αξιοποιήσει 8192 επεξεργαστικούς πυρήνες στον υπερυπολογιστή NERSC Cori.

1.4.1 Σύντομη παρουσίαση της γλώσσας Julia

Στην πράξη, η Julia καθιστά εύχρηστο ένα μικτό προγραμματιστικό μοντέλο προστακτικού και συναρτησιακού προγραμματισμού. Η δηλωτική φύση του συναρτησιακού προγραμματισμού μένει πιστή στη μαθηματική διατύπωση της θεωρίας και αυξάνει σημαντικά την εκφραστικότητα του κώδικα, προτρέποντας την αποσύνθεση πολύπλοκων ορισμών σε απλούστερους. Απελευθερώνει δε το πρόγραμμα από την έμμεση σειριοποίηση που επιβάλλει ο προστακτικός προγραμματισμός και διευκολύνει την αναδιάταξη του κώδικα, όπως και την κλιμάκωση της εκτέλεσής του σε περισσότερους υπολογιστικούς πόρους («παραλληλοποίηση»). Ο προστακτικός προγραμματισμός οδηγεί σε ορισμένες περιπτώσεις σε πιο φυσική ή εύκολη διατύπωση της λύσης, ενώ σε άλλες λειτουργεί υπό μορφή βελτιστοποίησης.

Ο σχεδιασμός της Julia προσπαθεί να διευκολύνει τον προγραμματιστή, αν επιθυμεί να υλοποιήσει τη λύση του με τον πιο πρόχειρο τρόπο, και να του επιτρέψει να τη βελτιώσει και να την κάνει αποδοτική με περισσότερη προσπάθεια και φροντίδα. Πιστή σε αυτήν την αρχή, υιοθετεί προαιρετικό σύστημα τύπων (δε θα ήταν άστοχη η σύγκριση με typed λ-calculus).

Σύμβολα και τιμές

Ένα πρόγραμμα Julia είναι κατά κανόνα ένα σύνολο ορισμών τύπων και τιμών. Στους τύπους και στις τιμές μπορεί να αντιστοιχηθούν ονόματα, σύμβολα. Όταν δηλώνεται

```
julia> x= 1
1
```

η τιμή 1 αντιστοιχίζεται στο σύμβολο x. Έτσι ο τύπος του x είναι ο τύπος της τιμής του:

```
julia> x|> typeof
Int64

julia> x= 1.0
1.0

julia> x|> typeof
Float64
```

Σε όλην αυτήν την παρουσίαση δε χρησιμοποιήθηκε ο όρος «μεταβλητή». Το x είναι απλώς ένα σύμβολο, που εδώ αντιστοιχίστηκε σε μια σταθερή, αμετάβλητη τιμή, και μετά *επαναντιστοιχίστηκε* σε μια διαφορετική σταθερή, αμετάβλητη τιμή.

Το x θα μπορούσε να αντιστοιχιστεί και σε μια όντως μεταβλητή τιμή, δηλαδή μία τιμή που επιτρέπεται να τροποποιηθεί:

```
julia> x= rand(Int8,5)
5-element Array{Int8,1}:
 31
  5
 86
 -3
 59

julia> x[5]= 0
0

julia> x
5-element Array{Int8,1}:
 31
  5
 86
 -3
  0
```

Εξίσου, η τιμή που συμβολίζει το x μπορεί να είναι ένας τύπος

```
julia> x= Int64
Int64

julia> x|> typeof
DataType
```

ή μια συνάρτηση

```
julia> x= (i)-> i+1
#3 (generic function with 1 method)

julia> x|> typeof
getfield(Main, Symbol("##3#4"))
```

```
julia> x(5)
6
```

Συναρτήσεις και μέθοδοι

Ο μεταγλωττιστής εσωτερικά αποδίδει τύπο σε κάθε τιμή που απαντά στο πρόγραμμα. Ο προγραμματιστής δεν απαιτείται να συσχετίσει ευθέως σύμβολα με τύπους. Μπορεί όμως, αν θέλει, να χρησιμοποιήσει τύπους για ένα βασικό σκοπό: **«πολλαπλή αποστολή»** (multiple dispatch) μεθόδων συνάρτησης βάσει τύπων. Οι όροι «μέθοδος» και «συνάρτηση» σημαίνουν διαφορετικά πράγματα στη Julia. Παραπάνω ορίσαμε μία ανώνυμη συνάρτηση που υλοποιείται από μία μέθοδο. Θα μπορούσαμε όμως να ορίσουμε και συνάρτηση που να μην υλοποιείται από καμία μέθοδο:

```
julia> function myfun end
myfun (generic function with 0 methods)

julia> myfun()
ERROR: MethodError: no method matching myfun()
```

Ας προσθέσουμε 2 μεθόδους στη συνάρτηση, για να φανεί η «πολλαπλή αποστολή» κι η χρήση τύπων:

```
julia> myfun(i::Int)= print("I'm an Int")
myfun (generic function with 1 method)

julia> myfun(3)
I'm an Int

julia> myfun(i::Float64)= print("I'm a Double!")
myfun (generic function with 2 methods)

julia> myfun(3.0)
I'm a Double!

julia> myfun(3)
I'm an Int

julia> myfun()
ERROR: MethodError: no method matching myfun()
```

Μέθοδος λοιπόν είναι ο συνδυασμός μίας συνάρτησης και μιας πλειάδας (tuple) ορισμάτων. Αυτή η σχεδίαση επιτρέπει έναν πολυμορφισμό συγκρίσιμο με της C++.

Σύνθετοι τύποι και παράμετροι

Είναι συχνό να θέλουμε να εκφράσουμε τιμές που συντίθενται από απλούστερες τιμές. Όπως το «struct» σε άλλες γλώσσες, έτσι και στη Julia μπορούμε να δημιουργήσουμε σύνθετους τύπους για να περιγράψουμε τέτοιες τιμές με τη λέξη-κλειδί struct:

```
julia> struct Point
  x::Int
  y::Int
end

julia> Point(2,3)|> typeof
Point

julia> Point(2,3).x
2
```

Για κάθε ονοματισμένο τύπο αυτόματα δηλώνεται και μία συνάρτηση με το ίδιο όνομα και μία μέθοδο, που παίρνει τόσα ορίσματα όσα τα στοιχεία του τύπου και αντιστοιχίζει με τη σειρά τις τιμές τους στην τιμή του σύνθετου τύπου που δημιουργεί.

Ο ορισμός του τύπου `Point` παραπάνω όμως μας περιορίζει στις 2 διαστάσεις. Πώς θα ορίζαμε έναν τύπο Point N διαστάσεων; Με παράμετρο τύπου:

```
julia> struct Point{N}
  coords::NTuple{N,Int}
end

julia> Point((3,4,5))
Point{3}((3, 4, 5))
```

Οι παράμετροι τύπου ορίζουν μία παραπάνω τιμή που είναι διαθέσιμη για χρήση στο πλαίσιο ορισμού της (ορισμός συνάρτησης ή τύπου). [1] Έχει όμως τη βασική διαφορά με άλλες τιμές να είναι προσπελάσιμη τη στιγμή της μεταγλώττισης. Έτσι, μπορεί να συμμετέχει στην περιγραφή του ίδιου του τύπου. Παραπάνω, ο τύπος `Point{3}` είναι διαφορετικός από πχ `Point{4}`.

Θα μπορούσαμε να κάνουμε το N-διάστατο τύπο Point πολύ πιο εύχρηστο, ορίζοντας περισσότερες συναρτήσεις, αλλά αυτό θα μας απασχολήσει αργότερα.

Unicode στον κώδικα

Ένα ακόμα βοηθητικό στοιχείο της γλώσσας είναι η χρήση γλύφων Unicode (UTF-8) στον κώδικα. Μπορούμε να χρησιμοποιήσουμε τους περισσότερους γλύφους Unicode στα ονόματα συμβόλων. Κάποιοι επιτρέπονται στα σύμβολα οποιασδήποτε τιμής (χαρακτηριστικά, όσους δείκτες ή εκθέτες προσδιορίζονται στη Unicode, που, δυστυχώς, δεν είναι το σύνολο) και κάποιοι άλλοι ορίζουν υποχρεωτικά τελεστές:

```
julia> ᶜx= 5
5

julia> ⊖(a,b)= a—b > 0 ? a—b : 0
⊖ (generic function with 1 method)
```

[1] Προς αποφυγίν σύγχυσης, οι «παράμετροι τύπου» δε χρησιμοποιούνται μόνο κατά τον ορισμό σύνθετων τύπων, αλλά και στον ορισμό μεθόδων συναρτήσεων.

```
julia> 3 ⊖ ᶜx
0

julia> ᶜx ⊖ 3
2
```

Η δυνατότητα αυτή θα διευκολύνει τη διατύπωση μαθηματικών ορισμών.

Ο αναγνώστης θα κρίνει ο ίδιος την αποτελεσματικότητα των παραπάνω στον κορμό της εργασίας, όπου θα έρθει σε επαφή με αυτό το ιδίωμα γραφής.

Κεφάλαιο 2

Η θεωρία HTM

2.1 Το πρόβλημα της πρόβλεψης ακολουθιών

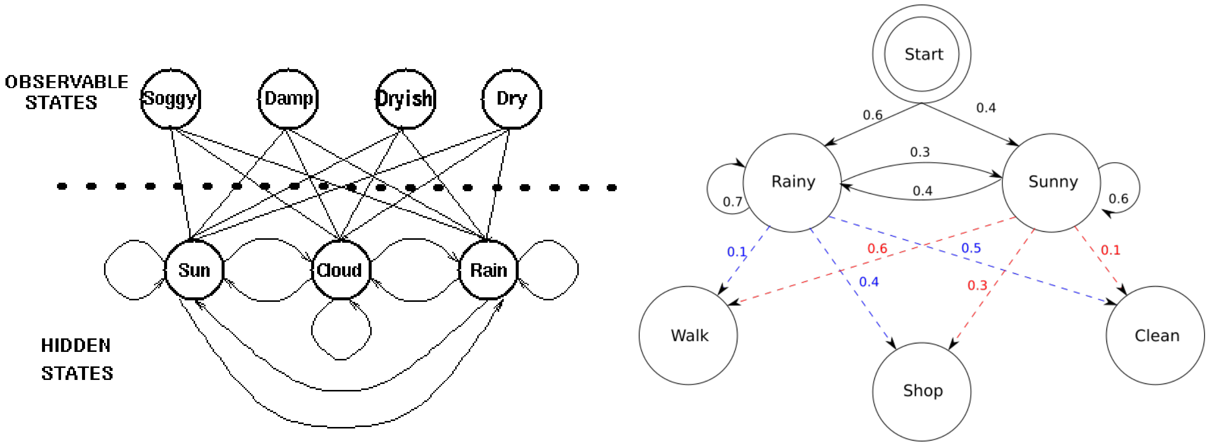
Τα σύνορα της τεχνητής νοημοσύνης εκτείνονται πέρα από το παραδοσιακό πρόβλημα της εκμάθησης ενός στατικού συνόλου δεδομένων με επιβλεπόμενες μεθόδους, όπως γίνεται σε πολλές σύγχρονες εφαρμογές [29]. Η τεχνητή νοημοσύνη καλείται σήμερα να αξιοποιήσει τις καταιγιστικές ροές δεδομένων που παρέχει το (ανθρωπογενές και μη) περιβάλλον, σε πραγματικό χρόνο, και δίχως την πολυτέλεια της προσήμανσης που απαιτεί η επιβλεπόμενη μάθηση — εφαρμογές Internet of Things που απαιτούν δυναμική αλληλεπίδραση με το περιβάλλον τους έρχονται στο μυαλό [1].

Επομένως, καλούμαστε να μοντελοποιήσουμε έναν κόσμο που αλλάζει [32]. Το μοντέλο οφείλει ή να είναι γενικότερο από όλες τις δυνατές μεταβολές του κόσμου ή να αλλάζει μαζί του. Η έννοια της ροής δεδομένων που αναφέρθηκε υπονοεί την έννοια του χρόνου, της αλληλουχίας, που επιτρέπει στο μοντέλο να αντλήσει πληροφορία από την αιτιώδη σχέση. Την ερμηνεία του κόσμου με τη βοήθεια της συνεπαγωγής και της αναγνώρισης απαιτήσεων και συνεπειών.

Ορίζεται έτσι το πρόβλημα της εκμάθησης και πρόβλεψης ακολουθιών. Δηλαδή, ο πράκτορας που παρακολουθεί μια αλληλουχία δεδομένων (γεγονότων) καλείται να προβλέψει τη συνέχεια και να δράσει έτσι, ώστε βέλτιστα να ανταμειφθεί. Η ανταμοιβή, το κίνητρο και γενικότερα ο στόχος έχουν τεθεί από εξωτερικό παράγοντα και βρίσκονται εκτός του πλαισίου του προβλήματος. Άμεση εφαρμογή της πρόβλεψης είναι και η αναγνώριση ανωμαλιών.

Η εκμάθηση ακολουθιών είναι κλασικό πρόβλημα, για το οποίο έχουν αναπτυχθεί κλασικές λύσεις. Κυριότερο και ευρύτερα διαδεδομένο είδος μοντέλων, ειδικά πριν τη σύγχρονη επάνοδο των νευρωνικών δικτύων, είναι τα Hidden Markov Models (2.1). Από το πεδίο των κλασικών νευρωνικών δικτύων προσφέρεται η λύση των νευρωνικών δικτύων χρονικής καθυστέρησης (TDNN). Η ουσιαστική συνεισφορά των κλασικών νευρωνικών δικτύων επιτυγχάνεται όμως με τα ανάδρομα δίκτυα (RNN), που γενίκευσαν τις παλαιότερες προσθιοδρομικές (feedforward) αρχιτεκτονικές ακριβώς για να μπορούν να αντιμετωπίσουν εγγενώς προβλήματα ακολουθιών. Ιδιαίτερης μνείας χρήζει η δομή «μακράς βραχυπρόθεσμης μνήμης» (LSTM). Από το 2015 έχει χρησιμοποιηθεί με επιτυχία σε ποικίλες εφαρμογές πολύ πιο σύνθετες από την εισαγωγή που παρουσιάζεται εδώ, όπως ως τεχνητή νοημοσύνη που παίζει το παιχνίδι StarCraft 2 [AlphaStar 42].

Σε αυτόν το χώρο λύσεων υπάρχουν παρόλα αυτά σημεία για βελτίωση. Οι περισσότερες εφαρμογές, συμπεριλαμβανομένου του AlphaStar, βασίζονται σε επιβλεπόμενη και ενισχυ-



Σχήμα 2.1: Hidden Markov Model

τική μάθηση, αφήνοντας ευρύ πεδίο εξερεύνησης για μη επιβλεπόμενα μοντέλα. Καθώς ο όγκος των ροών δεδομένων αυξάνεται εκθετικά [38], παρατηρείται αυξανόμενη ζήτηση για αλγορίθμους που προσαρμόζονται γρήγορα στις εξελισσόμενες στατιστικές των δεδομένων τους (συνεχής μάθηση), έχοντας πρόσβαση μόνο σε μικρό χρονικό παράθυρο πληροφορίας. Στην προσπάθεια αυτή αναπτύσσονται ενεργά τεχνικές για βελτίωση της αποδοτικότητας δειγμάτων της μάθησης [όπως 39], ή για παράκαμψη του προβλήματος, όπως η μεταφορική μάθηση (transfer learning) [41].

Στην ανάλυση των διαφόρων τρόπων ορισμού της τεχνητής νοημοσύνης, ο Wang [14] αναγνωρίζει τη μέθοδο «από τη δομή» για την έρευνα σε συστήματα που εμπνέονται ή μιμούνται το βασικό παράδειγμα ευφυούς συστήματος που επιλύει διαρκώς το παραπάνω πρόβλημα: τον εγκέφαλο, και ειδικά το φλοιό. Παρατηρείται μια τάση στο χώρο των τεχνητών νευρωνικών δικτύων ανασκόπησης της επαφής τους με τη βιολογική πραγματικότητα τα τελευταία χρόνια, όπως στο [19]. Το πιο ηχηρό παράδειγμα είναι ο μηχανισμός κάψουλας που πρότειναν ο Hinton και συνεργάτες [37], [40].

Σε αυτό λοιπόν το πλαίσιο, η μελέτη της HTM κρίνεται ιδιαίτερα καίρια.

2.2 Στοιχεία της Hierarchical Temporal Memory

Παρακάτω θα αναφερθούμε πάλι σε στοιχεία νευροεπιστήμης. Καθώς ο σκοπός της περιγραφής είναι η κατανόηση των αλγορίθμων HTM, πρέπει να γίνει αποδεκτή μια παρέκκλιση από την αυστηρότητα, χάριν απλότητας. Σε ό,τι βιολογικό στοιχείο αναφερθεί, ο αναγνώστης ας έχει υπόψιν ότι η πραγματικότητα είναι πάντα πιο πολύπλοκη και ότι εδώ παρουσιάζονται *χρήσιμες προσεγγίσεις*.

Η κεντρική θέση στην οποία βασίζεται η Hierarchical Temporal Memory τμηματοποιεί τον εγκεφαλικό φλοιό σε ένα ψηφιδωτό βασικών μονάδων επεξεργασίας, των **«φλοιικών στηλών» (cortical columns)**, που έχουν την ίδια δομή καθεμία και εκτελούν τους **ίδιους αλγορίθμους, αλλά σε διαφορετικά δεδομένα**. Η ιδέα αυτή έχει μακρά ιστορία στη νευροεπιστήμη, με μια συγκεντρωτική επισκόπηση από Defelipe, Markram κ.ά [15] να την οριοθετεί και να παρουσιάζει την ευρεία χρήση και κακομεταχείριση του όρου. Για παράδειγμα, η εισαγωγική πρόταση περί «ψηφιδωτού» παραπάνω πρέπει να αντιμετωπιστεί μόνο ως πρώτη προσέγγιση, γιατί οι στήλες δε φέρονται να εχουν γεωμετρικά σαφή σύνορα μεταξύ τους, αλλά διάχυτες ζώνες μετάβασης. Πρεσβευτής και βασικός αποκρυσταλλωτής της ιδέας είναι ο Mountcastle [8], με την ιδέα να χαίρει τόσο αποδοχής [26], όσο και κριτικής ως προς τη χρησιμότητά της [11]. Παρόλα αυτά, εδώ θα την υιοθετήσουμε. Έτσι, ο εγκεφαλικός φλοιός αποδομείται σε πλειάδα μονάδων επεξεργασίας κοινής αρχής, και οι αλγόριθμοι που θα παρουσιαστούν περιγράφουν τη λειτουργία της μονάδας.

Η φλοιική στήλη είναι λοιπόν ένας **πληθυσμός νευρώνων με κοινή συνδεσμολογία**: *λαμβάνουν το σήμα εισόδου από κοινές πηγές και στέλνουν το σήμα εξόδου σε κοινούς παραλήπτες*. Συχνά, τέτοιοι παραλήπτες είναι άλλες φλοιικές στήλες. Συντάσσονται έτσι επεξεργαστικές **ιεραρχίες**, με τα πρώτα στάδια της ιεραρχίας να δημιουργούν απλούστερα μοντέλα για τον κόσμο από τα μετέπειτα (κυρίως γιατί τα μετέπειτα συγκεντρώνουν περισσότερη πληροφορία).

Οι νευρώνες στους οποίους αναφερόμαστε παραπάνω, οι πυραμιδικοί νευρώνες, βρίσκονται κάθε στιγμή σε 1 από 2 καταστάσεις: ενεργοί ή ανενεργοί. Η ενεργοποίησή τους («δυναμικό δράσης») ερεθίζει άλλους νευρώνες με τους οποίους συνδέονται και μπορεί να τους οδηγήσει σε ενεργοποίηση. Μπορούμε λοιπόν να περιγράψουμε την κατάσταση του φλοιού κάθε στιγμή ως το σύνολο των νευρώνων που είναι ενεργό. Προκύπτει ότι το σύνολο αυτό είναι πολύ μικρό ποσοστό του συνολικού νευρικού πληθυσμού, δηλαδή η ενεργοποίηση είναι **αραιή**, περίπου 2%. Σημειώνεται ότι οι πυραμιδικοί νευρώνες είναι η μειοψηφία των νευρώνων στο φλοιό. Το βασικό κοινό τους χαρακτηριστικό είναι ότι στέλνουν τους άξονές τους μέσω της λευκής ύλης σε μακρινούς προορισμούς, για το οποίο και ονομάζονται κύριοι νευρώνες. Περισσότεροι είναι οι διάμεσοι νευρώνες (interneurons), των οποίων οι άξονες παραμένουν σε μικρή εμβέλεια, και θεωρείται ότι συμμετέχουν σε τοπικά κυκλώματα που εν τέλει ρυθμίζουν την ενεργοποίηση των κυρίων νευρώνων [13].

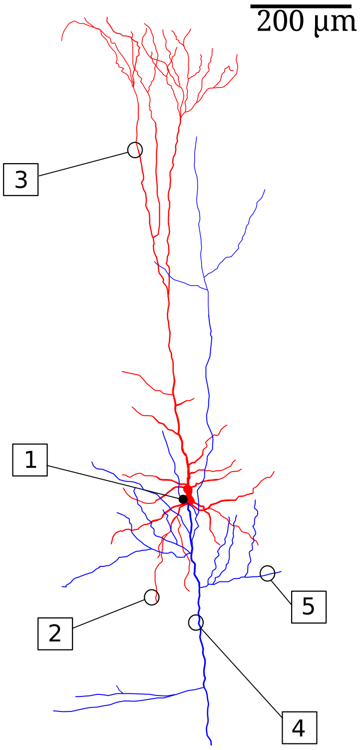
Το μοτίβο ενεργοποίησης ενός πληθυσμού νευρώνων της ίδιας στήλης αποτελεί στο πλαίσιο της θεωρίας HTM τη δομή δεδομένων του εγκεφάλου και ονομάζεται **αραιή διανεμημένη αναπαράσταση (SDR)**. Τα SDR αναλύονται στην ενότητα 2.2.2. Κάθε αίσθηση στέλνει SDR στις στήλες που την επεξεργάζονται·κάθε στήλη στέλνει SDR στους μύες ή σε άλλες στήλες ως έξοδο. Η είσοδος σε ένα μοντέλο HTM πρέπει επομένως να μεταφράζει τη φυσική ποσότητα που θέλουμε να επεξεργαστούμε σε SDR, με τη διαδικασία να ονομάζεται **κωδικοποίηση**. Αντίστοιχα, η ερμηνεία ενός SDR εξόδου ονομάζεται **αποκωδικοποίηση**.

Από τους κοινούς αλγορίθμους που υλοποιεί κάθε φλοιική στήλη, η HTM αυτή τη στιγμή περιγράφει 2: τη **χωρική συγκέντρωση** και τη **χρονική μνήμη**. Η χρονική συγκέντρωση είναι επίσης διαδικασία που υποτίθεται, αλλά δεν έχει περιγραφεί επαρκώς και αποτελεί βασικό σημείο για περαιτέρω μελέτη.

2.2.1 Μοντέλο νευρώνα

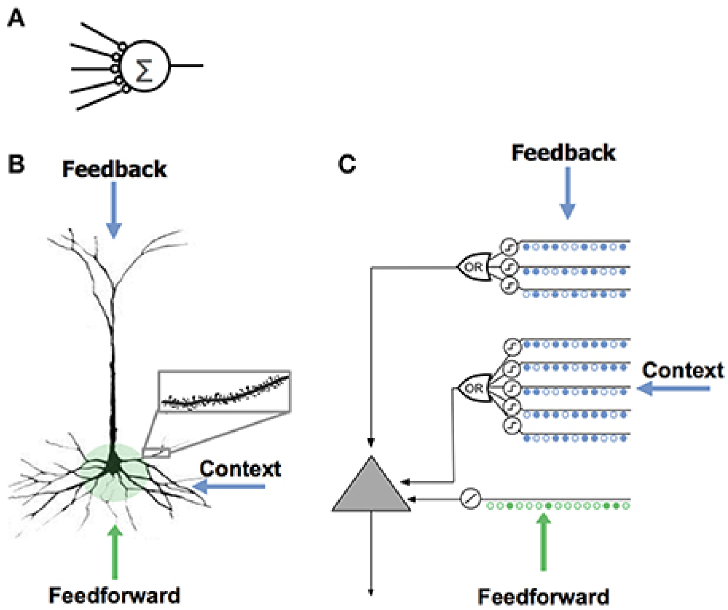
Νευρώνες

Στον ανθρώπινο εγκέφαλο υπάρχουν πολλά είδη νευρώνων, που διαφέρουν στη μορφολογία, στις ηλεκτρικές και χημικές τους ιδιότητες [22]. Η μόνη κατηγορία νευρώνα στην οποία βασίζεται το νευρικό μοντέλο της HTM είναι οι πυραμιδικοί νευρώνες 2.2 (το αποτέλεσμα της συμπεριφοράς άλλων νευρώνων συμπεριλαμβάνεται έμμεσα στη λογική των αλγορίθμων). Ο πυραμιδικός νευρώνας είναι ένα σύνθετο στοιχείο, με πολλούς χώρους και διανεμημένες λειτουργίες, που υλοποιούν τις λογικές πράξεις της άθροισης, του πολλαπλασιασμού, της χωρικής και χρονικής ολοκλήρωσης ταυτόχρονα και παράλληλα σε διαφορετικές ομάδες εισόδων. Η κεντρική δομή του νευρώνα, το σώμα, δέχεται ερεθίσματα από άλλους νευρώνες. Αρκετά τέτοια ερεθίσματα σε σύντομο χρονικό διάστημα



(α') Τυπική μορφή πυραμιδικού νευρώνα:
1. σώμα
2. βασικός δενδρίτης
3. κορυφαίος δενδρίτης
4. άξονας
5. παράπλευρος άξονας

[πηγή 43]



(β')
Α'. Μοντέλο τυπικού νευρώνα σε τεχνητό νευρωνικό δίκτυο
Β'. Αναπαράσταση πυραμιδικού νευρώνα
Γ'. Μοντέλο νευρώνα HTM

[πηγή 27]

Σχήμα 2.2: πυραμιδικός νευρώνας και μοντέλο νευρώνα στην HTM

είναι ικανά να ενεργοποιήσουν το νευρώνα. Οι πηγές των σημάτων που ο νευρώνας λαμβάνει στο σώμα του ονομάζονται υποδεκτικό πεδίο (receptive field) του νευρώνα. Από το σώμα φυτρώνουν οι εγγύς δενδρίτες. Ο ερεθισμός τους δεν είναι ικανός συνήθως να ενεργοποιήσει το νευρώνα, αρκεί όμως για να τον θέσει σε κατάσταση «επιφυλακής» (αποπολωμένος/προβλεπτικός), διευκολύνοντας τη μετέπειτα ενεργοποίησή του από σωματικά ερεθίσματα. Ομοίως και για τον ερεθισμό απομακρυσμένων ή κορυφαίων δενδριτών κατά μήκος του άξονα.

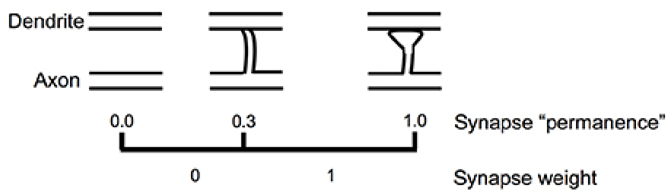
Στο πλαίσιο της HTM, νευρώνα ονομάζουμε μια δομή που δέχεται ερεθίσματα στους δενδρίτες της και τροποποιεί την τρισταθή της κατάσταση με βάση αυτά. Η κατάσταση μπορεί να είναι *ανενεργή, αποπολωμένη (αλλιώς προβλεπτική) ή ενεργή*. Όπως φαίνεται στο σχήμα 2.2β', ο νευρώνας HTM έχει 3 πύλες εισόδου. Ερεθίσματα στην προσθιοδρομική (feedforward) είσοδο προσμετρώνται και, αν ξεπεράσουν ένα κατώφλι, ο νευρώνας μεταβαίνει στην ενεργή κατάσταση. Ερεθίσματα στην είσοδο συμφραζομένων (context) είναι ικανά να προκαλέσουν αποπόλωση σε ανενεργό νευρώνα, αλλά όχι να τον ενεργοποιήσουν. Ομοίως και στην αναδρομική (feedback) είσοδο. Προσομοιώνονται έτσι μερικές συμπεριφορές του βιολογικού πυραμιδικού νευρώνα. Σε αντιπαραβολή, ο νευρώνας ενός κλασικού τεχνητού νευρωνικού δικτύου δεν έχει διακριτά λειτουργικά τμήματα. Αθροίζει μαζί όλα τα ερεθίσματα που λαμβάνει στη μία είσοδό του.

Συνάψεις

Η δομή που μεταφέρει σήματα μεταξύ νευρώνων ονομάζεται *σύναψη* και είναι κατά κανόνα **μονόδρομη**. Συνάψεις σχηματίζονται στο σημείο επαφής δύο νευρώνων, του προσυναπτικού και του μετασυναπτικού, συνήθως από τον άξονα του προσυναπτικού σε δενδρίτη ή στο σώμα του μετασυναπτικού. Στο φλοιό οι συνάψεις είναι χημικής φύσεως, όχι ηλεκτρικής. Όταν το δυναμικό δράσης του προσυναπτικού νευρώνα φθάσει σε σύναψη, προκαλεί την απελευθέρωση νευροδιαβιβαστών στον εξωκυττάριο χώρο, προς την κατεύθυνση του μετασυναπτικού νευρώνα. Ο μετασυναπτικός νευρώνας φέρει υποδοχείς που ανιχνεύουν την παρουσία νευροδιαβιβαστών και συμμετέχουν στην αποπόλωση ή υπερπόλωση του νευρώνα τους. Αυτός ο μηχανισμός εγγυάται τη μονόδρομη μεταγωγή σήματος δια μέσου της σύναψης. Εν τω βάθει ανάλυση του αντικειμένου παρουσιάζεται στο [16].

Οι συνάψεις χωρίζονται σε 2 κατηγορίες: διεγερτικές και ανασταλτικές. Η ενεργοποίηση μιας διεγερτικής σύναψης καθιστά πιο πιθανή την πρόκληση δυναμικού δράσης στο μετασυναπτικό νευρώνα, ενώ η ενεργοποίηση ανασταλτικής σύναψης την καθιστά λιγότερο πιθανή. Ένας τύπος προσυναπτικού νευρώνα δημιουργεί έναν τύπο συνάψεων: οι πυραμιδικοί δημιουργούν διεγερτικές και οι περισσότεροι διάμεσοι νευρώνες ανασταλτικές. Ο ίδιος νευρώνας όμως δέχεται σήματα και από διεγερτικές και ανασταλτικές συνάψεις, που ανταγωνίζονται μεταξύ τους. Οι διεγερτικές συνάψεις φέρουν συγκεκριμένους νευροδιαβιβαστές, με πιο συχνό το γλουταμικό. Οι ανασταλτικές χρησιμοποιούν διαφορετικούς, όπως το GABA.

Συνάψεις υπάρχουν σε όλες τις ζώνες ερεθισμού που αναφέρθηκαν: στο σώμα, στους εγγύς και κορυφαίους δενδρίτες. Οι συνάψεις είναι επίσης ο στόχος των μηχανισμών μάθησης/προσαρμογής. Χαρακτηρίζονται από μία μεταβλητή, ένα μέγεθος, που προσαρμόζεται με κανόνες που εμπίπτουν στον όρο «Hebbian learning», συγκεκριμένα, «πλαστικότητα εξαρτώμενη από το χρονισμό των ερεθισμάτων» (spike timing-dependent plasticity, STDP) [10]. Στο πλαίσιο της HTM όμως το μέγεθος **δε συνεπάγεται συναπτικό βάρος**: οι συνάψεις είναι συνδεδεμένες ή αποσυνδεδεμένες. Μια σύναψη με πολύ μικρό μέγεθος θεωρείται αποσυνδεδεμένη και δε μετάγει καθόλου ερεθίσματα. Καθώς μεγαλώνει, εφόσον περάσει ένα κατώφλι γίνεται συνδεδεμένη και μετάγει ερεθίσματα. Το μέγεθος δηλαδή της σύναψης κρίνει πόσο εύκολα η σύναψη θα συνδεθεί ή αποσυνδεθεί, και γι'αυτό ονομάζεται μονιμότητα.



Σχήμα 2.3: Μονιμότητα σύναψης

Ένας πυραμιδικός νευρώνας μπορεί να έχει χιλιάδες συνάψεις. Ελάχιστες από αυτές βρίσκονται κοντά στο σώμα και, όπως προαναφέρθηκε, μπορούν να τον ενεργοποιήσουν. Οι υπόλοιπες βρίσκονται στους δενδρίτες και, μεμονωμένα, έχουν πολύ μικρή επίδραση στο σώμα. Οι δενδρίτες όμως προκύπτει ότι παίζουν ρόλο σύνθετων επεξεργαστικών στοιχείων. Αν ενεργοποιηθούν *περισσότερες από 8-20 συνάψεις* σε μικρό χρόνο στον ίδιο δενδρίτη, τότε ο δενδρίτης ενεργοποιείται και δημιουργεί ένα δυναμικό που προκαλεί την αποπόλωση όλου του νευρώνα [27]. Καθώς ο δενδρίτης ενεργοποιείται μόνο αν συμπέσει χρονικά και χωρικά η ενεργοποίηση πολλών συνάψεων, λειτουργεί ουσιαστικά ως **«ανιχνευτής συμπτώσεων»** (coincidence detector).

Ας εξετάσουμε γιατί αρμόζει αυτός ο όρος, μελετώντας τα μοτίβα που ενεργοποιούν τους δενδρίτες στην επόμενη ενότητα.

2.2.2 Αραιές Διανεμημένες Αναπαραστάσεις (SDR)

Με βάση την παρατήρηση της αραιής ενεργοποίησης των νευρώνων στον εγκέφαλο σχεδιάζουμε ένα μεγάλο, αραιό, δυαδικό διάνυσμα που αναπαριστά την κατάσταση ενεργοποίησης κάθε νευρώνα της περιοχής που μελετούμε. Ουσιαστικά, αυτά τα αραιά διανύσματα αποτελούν τη «δομή δεδομένων» [6], [28] του εγκεφάλου.

Αναφέρθηκε στην 2.2.1 ότι οι δενδρίτες λειτουργούν ως ανιχνευτές συγκεκριμένων μοτίβων ενεργοποίησης. Πώς όμως αρκούν 8-20 συνάψεις για να διακρίνουν συγκεκριμένα μοτίβα μέσα σε μεγάλους νευρικούς πληθυσμούς; Το κλειδί είναι η αραιή ενεργοποίηση.

Έστω ένας πληθυσμός 200K νευρώνων, όπου ενεργοί είναι το 1%, και ένας δενδρίτης που απαιτεί 10 ερεθίσματα για να ενεργοποιηθεί. Αν τυγχάνει να έχει συνδεδεμένες συνάψεις σε 10 από τους 2000 ενεργούς νευρώνες, ενεργοποιείται. Το «μοτίβο» εν προκειμένω είναι το σύνολο των συγκεκριμένων 2000 νευρώνων που ενεργοποιήθηκαν. Προφανώς, αφού έχει συνάψεις μόνο με 10/2000 ενεργούς νευρώνες, ο δενδρίτης θα μπορούσε να ενεργοποιηθεί κατά λάθος και σε πολλά διαφορετικά μοτίβα, που τυχαίνει να μοιράζονται τους ίδιους 10 νευρώνες με το αρχικό. Πόση είναι η πιθανότητα ενός τέτοιου σφάλματος; 9.8×10^{-21}.

Παρακάτω θα μελετήσουμε τις ιδιότητες των SDR για να διαπιστωθεί πώς προκύπτει αυτό το αποτέλεσμα.

Ορισμοί SDR

Έστω N-bit SDR $s = 0, 1^N$. Ο αριθμός των μονάδων στο s $w = count(s)$ ονομάζεται πληθάριθμος. Η χωρητικότητα SDR με μέγεθος N και πληθάριθμο w είναι το πλήθος των διαφορετικών SDR με αυτή τη μορφή, δηλαδή οι συνδυασμοί των N ανά w:

$$\text{χωρητικότητα(N,w)} = \binom{N}{w} = \frac{N!}{w!(N-w)!}$$

Έστω 2 SDR A,B μήκους N. Ορίζουμε:

Union	$A\|B$
Overlap	$A\&B$
Overlap score	$\\|A\&B\\|$
Overlap set(θ)	{K όπου $\\|A\&K\\| > \theta$}
Ταιριάζουν(θ)	A *ταιριάζει*$_\theta$ $B \iff A, B \in$ ίδιο overlap set(θ)

Ταύτιση SDR και θόρυβος

Ας μελετήσουμε τον ορισμό ότι δύο SDR A με πληθάριθμο w και B *ταιριάζουν*. Έστω B:= A + 30% θόρυβος τυχαίας αναστροφής bit. Τότε το προσδοκώμενο overlap score A,B θα είναι $70\%w$. Αν $\theta := 70\%w$, τα A,B προσδοκάται να ταιριάζουν.

Το παράδειγμα αυτό μπορεί να εστιαστεί για την περίπτωση της ενότητας 2.2.2. Έστω A το SDR της ενεργοποίησης των 200K νευρώνων με w=2000 ενεργούς, και ο δενδρίτης που έχει συνάψεις με 30 από τους ενεργούς. Υπό την οπτική του δενδρίτη έχουμε το SDR $\hat{A}$ με μέγεθος 200K και $\hat{w} = 30$, γιατί οι υπόλοιποι ενεργοί νευρώνες δεν τον ερεθίζουν. Αν το όριο ενεργοποίησης του δενδρίτη είναι θ=20, τότε ακόμα και με 30% θόρυβο τυχαίας αναστροφής bit στον πληθυσμό των 200K νευρώνων αναμένουμε ο δενδρίτης να δεχθεί $\hat{w}_n = 70\%\hat{w} = 21 > \theta$ ερεθίσματα και επομένως να ενεργοποιηθεί, παρόλο το θόρυβο.

Θεωρήσαμε ότι το δεύτερο SDR είναι αποτέλεσμα θορύβου στο πρώτο κι επομένως ήταν θεμιτό ο δενδρίτης να τα ταυτίσει. Πόση όμως είναι η πιθανότητα να ενεργοποιηθεί ο ίδιος δενδρίτης από ένα διαφορετικό, τυχαίο SDR με πληθάριθμο $\hat{w}$ που δεν προκύπτει από το αρχικό; Σε αυτήν την περίπτωση η ενεργοποίηση θα ταύτιζε ψευδώς το τυχαίο SDR με το πρώτο. Η πιθανότητα ψευδούς ταύτισης είναι [1]:

$$p_{fp} = \frac{\|overlap\_set(\theta)\|}{\chi\omega\rho\eta\tau\iota\kappa\acute{o}\tau\eta\tau\alpha(N,w)} = \frac{\sum_{b=\theta}^{\hat{w}} \binom{\hat{w}}{b}\binom{N-\hat{w}}{\hat{w}-b}}{\binom{N}{w}} = \frac{8\text{e}53}{1\text{e}4862} = 5\text{e}{-}4809 \tag{2.1}$$

Εν προκειμένω, η μικροσκοπική πιθανότητα σφάλματος οφείλεται κυρίως στο γιγαντιαίο μέτρο του χώρου (N,w). Η πιθανότητα τυχαίας σύγκρουσης θα ήταν όμως αμελητέα και σε πολύ μικρότερο χώρο.

Καταδεικνύεται έτσι ότι η αναγνώριση SDR είναι μια διαδικασία *ανθεκτική στο θόρυβο*. Επίσης, αναφαίνεται μια *σχεδιαστική ελευθερία* που προσδίδει η χρήση SDR στο σύστημα: ανταλλαγή μεγέθους με ευρωστία στο θόρυβο.

Ταχεία ανάκληση SDR και παράλληλα ενδεχόμενα

Έστω ότι παρατηρούμε μια ακολουθία από SDR $\{s_t\}$. Κάποια χρονική στιγμή t_0 ρωτούμε αν έχουμε δει στη μέχρι τώρα ακολουθία $\{s_{0..t_0}\}$ το SDR Q. Πώς μπορούμε να απαντήσουμε γρήγορα σε αυτό το ερώτημα;

Εφόσον γινεται δεκτή μια πιθανότητα σφάλματος, που μπορεί να προσδιοριστεί ως προς τα μεγέθη και τον πληθάριθμο των SDR όπως παραπάνω, δε χρειάζεται να έχουμε αποθηκεύσει όλη την ακολουθία $\{s_{0..t_0}\}$ και να συγκρίνουμε κάθε στοιχείο με το Q. Αντίθετα, διατηρούμε μόνο την ένωση U των SDR που έχουν προηγηθεί και συγκρίνουμε το Q με το U.

Κάθε χρονική στιγμή,

$$U_t = U_{t-1} | s_t$$

όπου |: δυαδικό OR

Προφανώς,

$$B\ \textit{ταιριάζει}_\theta\ s_i \implies B\ \textit{ταιριάζει}_\theta\ U$$
$$B\ \textit{ταιριάζει}_\theta\ U \centernot\implies \exists i : B\ \textit{ταιριάζει}_\theta\ s_i$$

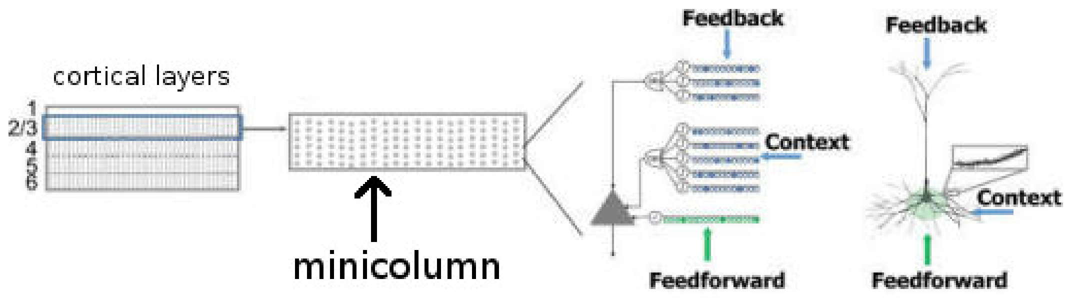
γιατί ενδέχεται τμήματα του B να ταιριάζουν με τμήματα από διαφορετικά s_i. Ακολουθώντας όμως τη λογική της προηγούμενης παραγράφου, για επαρκώς αραιά s_i, U η πιθανότητα σύγκρουσης είναι αρκετά μικρή. Οπότε μπορούμε να δεχθούμε το συμπέρασμα

$$B\ \textit{ταιριάζει}_\theta\ U \implies_p \exists i : B\ \textit{ταιριάζει}_\theta\ s_i$$

με μια πιθανότητα σφάλματος p.

Αν τα s_t είναι μεταξύ τους ανεξάρτητα και η αραιότητά τους προκύπτει από δειγματοληψία Bernoulli με πιθανότητα ανά θέση p, τότε το U_t επίσης προκύπτει από δειγματοληψία

[1] υπολογίστηκε με τον κώδικα του παραρτήματος Β′



Σχήμα 2.4: *Αποδόμηση φλοιικού επιπέδου και μικροστήλες.* Μία φλοιική στήλη του νεοφλοιού αποτελείται από 6 επίπεδα νευρώνων και φαίνεται αριστερά. Αν πάρουμε ένα από τα επίπεδα, θα έχει πάχος μερικών νευρώνων. Οι νευρώνες του ίδιου επιπέδου που είναι τοποθετημένοι κατακόρυφα ορίζουν (σε πρώτη προσέγγιση) μια μικροστήλη. [πηγή 25, (τροποποιημένο)]

Bernoulli, με πιθανότητα ανά θέση $1-(1-p)^t$. Ακόμα και για αρκετά αραιά s_t, καθώς το t αυξάνει το U γρήγορα γίνεται πυκνό. Καθώς γίνεται πυκνό, η πιθανότητα σφάλματος p αυξάνεται και το συμπέρασμα χάνει την αξία του. Επομένως, αυτή η μέθοδος είναι χρήσιμη μόνο όταν ο αριθμός των ανεξάρτητων s_t που συγχέουμε στο U είναι ελεγχόμενος.

Αν όμως ερμηνεύσουμε τα s_t ως πεπερασμένου αριθμού διακριτά ενδεχόμενα και το U είναι η δραστηριότητα ενός πληθυσμού νευρώνων, ο μηχανισμός αυτός επιτρέπει στον πληθυσμό να κωδικοποιεί τα διακριτά ενδεχόμενα παράλληλα και να μπορέσει, όταν έχει περισσότερη πληροφορία υπό τη μορφή συνθήκης S, να επιλύσει την αμφιβολία περιορίζοντας τη δραστηριότητά του στο $U\&S$

2.2.3 Μοντέλο δικτύου

Ο νεοφλοιός, όπως αναφέρθηκε στην ενότητα 1.3.2, δομείται συνήθως από 6 επίπεδα νευρώνων. Μία φλοιική στήλη χωρισμένη στα 6 επίπεδα φαίνεται στο σχήμα 2.4. Ορίσαμε τη φλοιική στήλη με βάση την κοινή συνδεσμολογία των νευρώνων της, καθώς τα ερεθίσματα έρχονται από και φεύγουν προς κοινές περιοχές. Έτσι παρουσιάσαμε τους νευρώνες της στήλης ως πληθυσμούς που κωδικοποιούν μία κοινή οντότητα — το βασικό τμήμα των εγκεφαλικών κυκλωμάτων. Είναι ακριβέστερο να μεταφέρουμε τον ορισμό αυτό από τη στήλη σε κάθε επίπεδο της στήλης. Μέσα σε μία στήλη δηλαδή τα 6 επίπεδα σχηματίζουν ξεχωριστά κυκλώματα, που όμως σχετίζονται μεταξύ τους και παίζουν διαφορετικούς λειτουργικούς ρόλους. Μία υπόθεση για τη συνδεσμολογία των επιπέδων στο τωρινό πλαίσιο της θεωρίας HTM φαίνεται στο σχήμα 2.5. Οι αλγόριθμοι που μελετούμε σε αυτήν την εργασία περιγράφουν κυρίως το επίπεδο 4 του σχήματος 2.5, που λαμβάνει είσοδο από τις αισθήσεις. Επίσης, εφεξής όταν γίνεται αναφορά σε *«επίπεδο»* θα εννοείται η τομή ενός επιπέδου και μιας στήλης, δηλαδή ένα στοιχειώδες κύκλωμα.

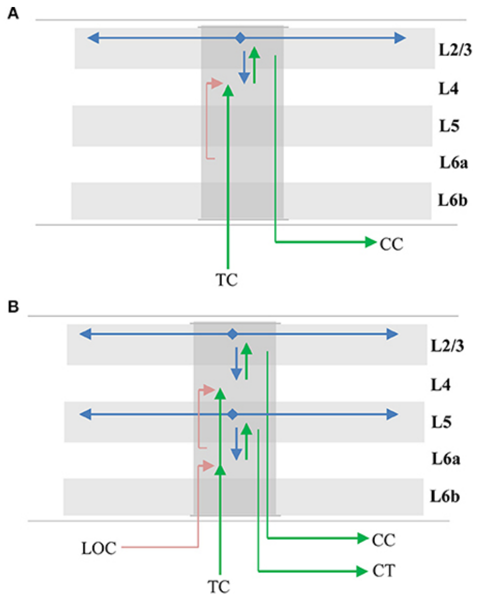
Μικροστήλες

Το ελάχιστο νευρικό κύκλωμα, τόσο στο νεοφλοιό, όσο και στην HTM, είναι η μικροστήλη (minicolumn). Σε μία μικροστήλη ανήκουν νευρώνες που λαμβάνουν ερεθίσματα στις εγγύς συνάψεις τους από τις ίδιες συνδέσεις μακρινής απόστασης (από τους ίδιους νευρώνες των περιοχών που στέλνουν είσοδο). Δηλαδή, οι νευρώνες μιας μικροστήλης μπορούμε να θεωρήσουμε ότι έχουν *ίδια είσοδο στις εγγύς συνάψεις τους και διαφέρουν μόνο ως προς την είσοδο στις απομακρυσμένες και κορυφαίες συνάψεις.* Για περισσότερες πληροφορίες στην οργάνωση των νευρώνων σε στήλες και μικροστήλες προτείνεται η θεμελιώδης εργασία των Markram et al «Reconstruction and simulation of neocortical microcircuitry» [22].

Καθώς οι εγγύς συνάψεις μεταφέρουν τη βασική πληροφορία που αναπαριστά ο πληθυσμός των νευρώνων ενός επιπέδου κι οι απομακρυσμένες τον ρυθμίζουν με βάση συμφραζόμενα, οι μικροστήλες επιτρέπουν σε μια μικρή ομάδα 10-20 νευρώνων να κωδικοποιεί *τα ίδια σύμβολα σε διαφορετικά συμφραζόμενα*. Όλοι οι νευρώνες της μικροστήλης μοιράζονται το ίδιο υποδεκτικό πεδίο, ενώ μεταξύ τους υπάρχει ανταγωνισμός.

Όπως αναφέρθηκε στην ενότητα 2.2.1, υπάρχουν τόσο διεγερτικές, όσο και ανασταλτικές συνάψεις, και κατ'επέκταση διεγερτικοί και ανασταλτικοί νευρώνες. Οι νευρώνες μίας μικροστήλης έχουν βρόχο ανάδρασης μέσω ενός ανασταλτικού διάμεσου νευρώνα. Αν κάποιος νευρώνας της μικροστήλης ενεργοποιηθεί λίγο νωρίτερα από τους υπόλοιπους, είναι ικανός να προκαλέσει την ενεργοποίηση του διάμεσου που τους εμπλέκει, ο οποίος με τη σειρά του να αποτρέψει την ενεργοποίηση όσων νευρώνων δεν έχουν προλάβει να ενεργοποιηθούν (σχήμα 2.6). Η ενεργοποίηση του νευρώνα είναι μια εύρωστη διαδικασία που υπόκειται σε θετική ανάδραση. Οπότε, αναστολή μετά την ενεργοποίηση δε θα επιφέρει ουσιαστικό αποτέλεσμα.

Έτσι υλοποιείται μια ανταγωνιστική αρχιτεκτονική («winner-takes-all»), όπου, αν κάποιοι νευρώνες είναι ήδη σε αποπολωμένη/προβλεπτική κατάσταση όταν έρθει προσθιοδρομική είσοδος που ερεθίζει το minicolumn, θα ενεργοποιηθούν πρώτοι και θα αποτρέψουν την ενεργοποίηση των υπολοίπων.



Σχήμα 2.5: Επίπεδα μέσα σε μία στήλη νεοφλοιού. Το μοντέλο συνδεσμολογίας που επισημαίνεται είναι τρέχουσα υπόθεση της θεωρίας HTM.

- Προσθιοδρομική ροή ερεθισμάτων (εγγύς συνάψεις)
- συμφραζόμενα (απομακρυσμένες/κορυφαίες συνάψεις)
- σήμα θέσης (απομακρυσμένες συνάψεις, εκτός πεδίου μελέτης αυτής της εργασίας)

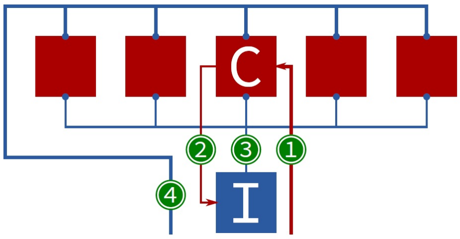
[πηγή 36]

Μεταξύ μικροστηλών

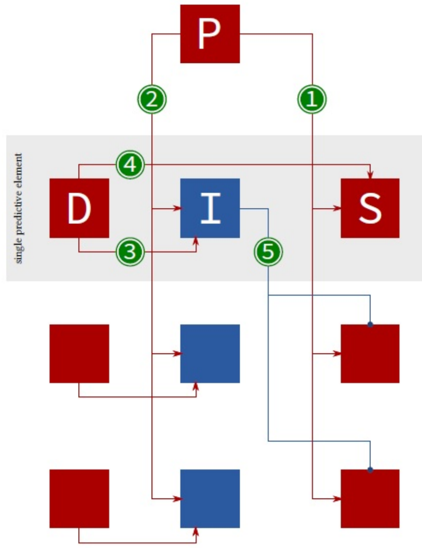
Φαινόμενα ανταγωνισμού μέσω αμοιβαίας αναστολής προκύπτουν και μεταξύ γειτονικών μικροστηλών, με τη βοήθεια άλλων διάμεσων νευρώνων (σχήμα 2.6). Το αποτέλεσμα είναι να εξασφαλίζεται ότι σε κάθε γειτονιά μόνο ένα αραιό υποσύνολο των πιο έντονα ερεθισμένων μικροστηλών θα επικρατήσει και θα ενεργοποιηθεί, σιγώντας τις υπόλοιπες. Ονομάζουμε το μηχανισμό αυτό «τοπική αναστολή». Στους αλγορίθμους που ακολουθούν, αντίθετα με τον εγκέφαλο, δεν είναι απαραίτητο να θεωρήσουμε ότι σχηματίζονται τοπικά γειτονιές·αντίθετα, μπορούμε να θεωρήσουμε ολόκληρο το επίπεδο μία γειτονιά, μηχανισμό που αποκαλείται «ολική αναστολή».

2.2.4 Κωδικοποιητές & αποκωδικοποιητές

Ο εγκέφαλος έχει ως βασικές διεπαφές με τον έξω κόσμο τον αμφιβληστροειδή χιτώνα του ματιού και τον κοχλία του αυτιού. Και τα 2 αυτά συστήματα μπορούμε να θεωρήσουμε



(α΄) Υλοποίηση χρονικού συγκεντρωτή με βάση το χρονισμό. Κάθε κύτταρο C αντιπροσωπεύει μία μικροστήλη. Δέχεται προσθιοδρομικά ερεθίσματα από τη ροή (1). Ο χρόνος που απαιτείται για την ενεργοποίηση της μικροστήλης εξαρτάται από το πόσα ερεθίσματα λαμβάνει·μια μικροστήλη με πολλά ερεθίσματα ενεργοποιείται ταχύτερα. Ερεθιζόμενος με τη ροή (2), ο ανασταλτικός νευρώνας I ενεργοποιείται μετά την ενεργοποίηση προκαθορισμένου αριθμού μικροστηλών. Με τη ροή (3) ο I αποτρέπει την ενεργοποίηση περισσότερων μικροστηλών.



(β΄) Υλοποίηση μιας μικροστήλης. Τα 3 στοιχεία D,I,S (δενδρίτης, αναστολέας, σώμα) μαζί περιγράφουν 1 νευρώνα. Το στοιχείο P είναι μοναδικό ανά μικροστήλη και είναι το ίδιο με το στοιχείο C του σχήματος (α΄). Με τη ροή (1) προκαλεί την ενεργοποίηση όλων των σωμάτων της μικροστήλης. Όμως ο δενδρίτης (εδώ μόνο 1/κύτταρο) συγκεντρώνει ερεθίσματα από την προηγούμενη χρονική στιγμή και, αν ενεργοποιηθεί, προκαλεί αποπόλωση του δικού του S και I με τις ροές (3),(4). Σε αυτήν την περίπτωση, αν στην επόμενη χρονική στιγμή το P ενεργοποιηθεί, θα ενεργοποιήσει το I, το οποίο με τη ροή (5) θα αναστείλει τα υπόλοιπα σώματα της μικροστήλης.

Σχήμα 2.6: Κυκλωματικές υλοποιήσεις μικροστηλών στο HICANN [23]. Προσφέρουν εποπτεία της οργάνωσης των μικροκυκλωμάτων μέσω του μηχανισμού της *αναστολής*. Σε σχέση με τη θεωρία HTM εφαρμόζουν κάποιες απλοποιήσεις, όπως μόνο 1 δενδρίτη ανά κύτταρο. [πηγή 20]

ότι μετατρέπουν τα φυσικά ερεθίσματα του κόσμου σε μια πρώτη μορφή SDR, για να ερεθίσουν με τη σειρά τους το επόμενο επίπεδο νευρώνων. Το ρόλο αυτό στα συστήματα HTM παίζουν οι κωδικοποιητές.

Οι κωδικοποιητές μπορεί να είναι από πολύ απλά έως εξαιρετικά πολύπλοκα συστήματα (βλέπε αμφιβληστροειδή). Ενας απλός κωδικοποιητής μπορεί να μετατρέπει ακέραιους αριθμούς συγκεκριμένου εύρους σε SDR. Ένας πιο σύνθετος μπορεί να παράγει SDRs από τη γεωγραφική θέση ενός αισθητήρα GPS [30]. Μπορεί ακόμα να σχεδιαστεί κωδικοποιητής για λέξεις ή κείμενα, όπως κάνει η εταιρία cortical.io [21].

Ένας επιτυχημένος κωδικοποιητής πρέπει να ακολουθεί ορισμένες βασικές αρχές [30]:

1. Είσοδοι με σημασιολογική ομοιότητα πρέπει να παράγουν SDRs με μεγάλη επικάλυψη
2. Η ίδια είσοδος πρέπει να παράγει πάντα το ίδιο SDR
3. Για κάθε είσοδο τα SDRs πρέπει να έχουν ίδιο μέγεθος
4. Για κάθε είσοδο τα SDRs πρέπει να έχουν κοντινό πληθάριθμο, αρκετά υψηλό για να

προσδίδει ανθεκτικότητα στο θόρυβο

Αντίστοιχα με τους κωδικοποιητές, πρακτικές εφαρμογές απαιτούν και αποκωδικοποιητές. Ένας πρακτικός αποκωδικοποιητής θα παρουσιαστεί στην υλοποίηση της HTM. Βασίζεται στην παρατήρηση ότι η δραστηριότητα της χρονικής μνήμης είναι η απεικόνιση ενός συμβόλου – εισόδου στο πλαίσιο των χρονικών του συμφραζομένων. Ο αποκωδικοποιητής που χρησιμοποιείται εδώ είναι ένα απλό ενός επιπέδου τεχνητό νευρωνικό δίκτυο που μαθαίνει να συσχετίζει τη δραστηριότητα της χρονικής μνήμης με τις εισόδους του κωδικοποιητή.

2.3 Αλγόριθμοι της Hierarchical Temporal Memory

Αυτή τη στιγμή στη θεωρία HTM περιγράφονται 2 βασικοί αλγόριθμοι. Ο πρώτος κανονικοποιεί τα SDR εισόδου και ονομάζεται *Χωρικός Συγκεντρωτής* (Spatial Pooler). Ο δεύτερος μαθαίνει ακολουθίες εισόδων, εμπλουτίζοντας την αναπαράσταση της εισόδου με τα χρονικά της συμφραζόμενα, και ονομάζεται *χρονική μνήμη* (Temporal Memory).

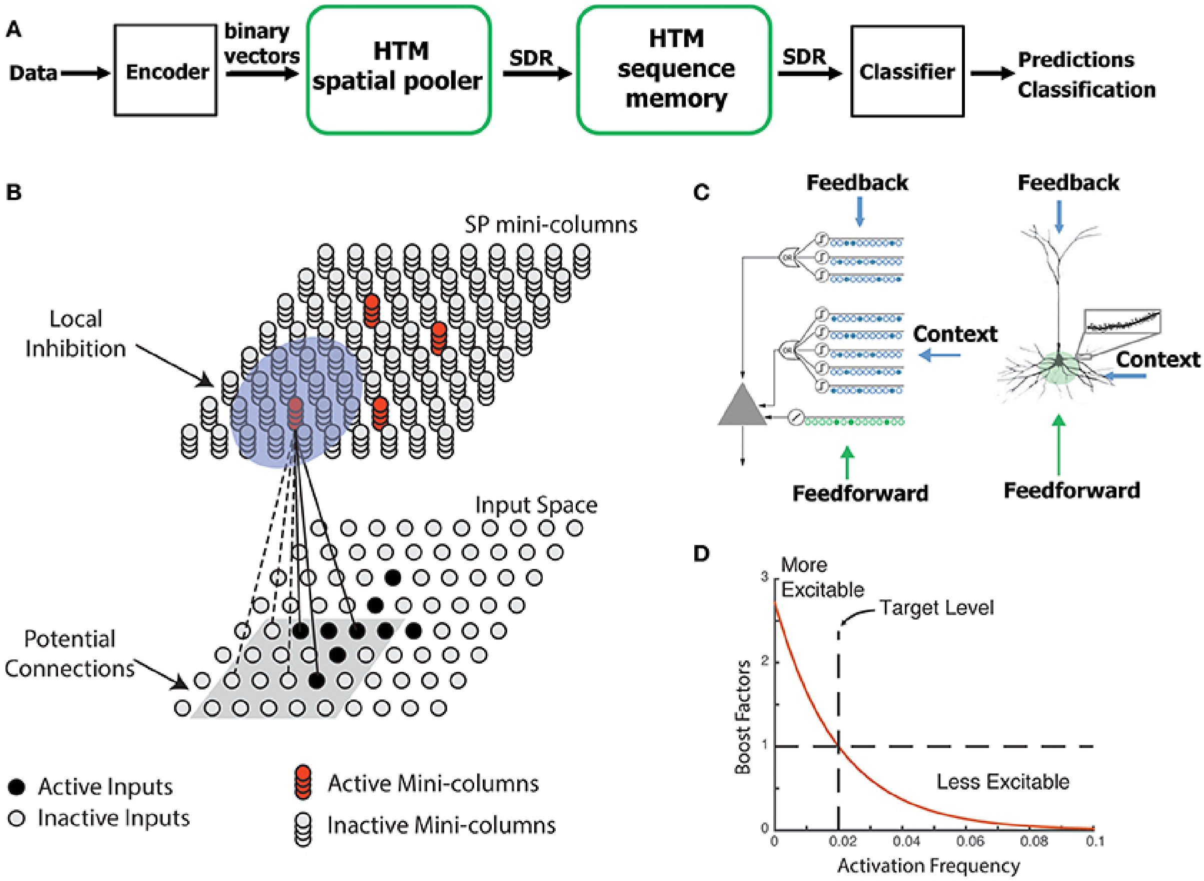
2.3.1 Χωρικός Συγκεντρωτής

Ο χωρικός συγκεντρωτής δουλεύει στο επίπεδο των μικροστηλών ενός επιπέδου, αξιοποιώντας τις εγγύς τους συνάψεις (προσθιοδρομική ροή) με το προηγούμενο ιεραρχικά επίπεδο (ή κωδικοποιητή) και το μηχανισμό αναστολής μεταξύ μικροστηλών που περιγράφηκε στην ενότητα 2.2.3. Ο αλγόριθμος αυτός περιγράφει πώς το SDR της εισόδου ενεργοποιεί τις μικροστήλες ενός επιπέδου μέσω του ερεθισμού των εγγύς συνάψεων και πώς αυτές προσαρμόζονται (μαθαίνουν) με εφαρμογή ενός κανόνα «πλαστικότητας εξαρτώμενης από το χρονισμό των ερεθισμάτων» (spike timing-dependent plasticity, STDP) [10]. Ο χωρικός συγκεντρωτής περιγράφεται αναλυτικά από τη Numenta στο [35].

Ο στόχος του αλγορίθμου είναι *να προετοιμάσει τα SDRs εισόδου για περαιτέρω επεξεργασία, εξασφαλίζοντας ορισμένες ιδιότητες*. Βασική αρχή λειτουργίας του είναι ότι είσοδοι που μοιάζουν μεταξύ τους (έχουν μεγάλη επικάλυψη) δημιουργούν εξόδους που μοιάζουν ανάλογα μεταξύ τους. Αυτή η αρχή ονοματίζει και τον αλγόριθμο.

Οι ιδιότητες που εξασφαλίζει ο χωρικός συγκεντρωτής στα SDR εξόδου είναι:

1. **Κανονικοποίηση της αραιότητας**.
2. **Ευρωστία στο θόρυβο**. Η έξοδος δε θα πρέπει να είναι ευαίσθητη στην παρουσία θορύβου στην είσοδο
3. **Ανεκτικότητα στα σφάλματα**. Αν καταστραφεί ή δε λειτουργεί σωστά μέρος των νευρώνων είτε της εισόδου είτε της εξόδου, η επίδραση στο συνολικό μηχανισμό πρέπει να είναι μικρή.
4. **Διαρκής προσαρμογή** στα εξελισσόμενα στατιστικά χαρακτηριστικά της εισόδου.
5. **Διασπορά της ενεργοποίησης** σε όλες τις μικροστήλες του συστήματος, εξασφάλιση υψηλής εντροπίας στην ενεργοποίηση.
6. Διατήρηση της τοπολογίας του χώρου εισόδου, εφόσον υπάρχει.



Σχήμα 2.7: Χωρικός Συγκεντρωτής HTM.

Α΄. Πλήρες σύστημα HTM για πρόβλεψη χρονοσειρών

Β΄. Εποπτική αναπαράσταση λειτουργίας του χωρικού συγκεντρωτή, υποθέτοντας τοπολογία (γενικά προαιρετική). Λόγω τοπολογίας, κάθε μικροστήλη έχει υποδεκτικό πεδίο ένα υποσύνολο (ολισθαίνον παράθυρο) του χώρου εισόδου, όπως φαίνεται από τη γκρίζα περιοχή. Η μπλε έλλειψη δείχνει τη γειτονιά των μικροστηλών στην οποία εφαρμόζεται τοπικός ανταγωνισμός μέσω αναστολής. Οι πλήρεις γραμμές δείχνουν συνάψεις που ισχυροποιούνται, ενώ οι διακεκομμένες, που εξασθενούν.

Γ΄. Οι συνάψεις που συμμετέχουν στο χωρικό συγκεντρωτή είναι οι εγγύς (προσθιοδρομικές)

Δ΄. Παρώθηση: ενίσχυση της ευαισθησίας μικροστηλών με ιστορικά σπάνια ενεργοποίηση

[πηγή 35]

Βέβαια, η ιδιότητα 5 είναι μάλλον προϋπόθεση για τις 2,3.

Αν και αναφέρθηκε ο «αλγόριθμος» του χωρικού συγκεντρωτή, θα ήταν ακριβέστερο να δούμε το χωρικό συγκεντρωτή ως ένα δυναμικό σύστημα διακριτού χρόνου, με εσωτερική κατάσταση, συνάρτηση αρχικοποίησης και βήματος/μετάβασης. Η βασική μεταβλητή κατάστασης του χωρικού συγκεντρωτή είναι οι εγγύς συνάψεις μεταξύ της προσυναπτικής εισόδου και των μετασυναπτικών μικροστηλών του. Ο «αλγόριθμος» αναφέρεται ακριβέστερα σε αυτόν που υλοποιεί η συνάρτηση βήματος.

Με την άφιξη των ερεθισμάτων από την είσοδο υπολογίζεται αρχικά η «επικάλυψη» (overlap) της εισόδου από κάθε μικροστήλη, δηλαδή το πλήθος ερεθισμάτων που μετήχθησαν μέσω συνδεδεμένων συνάψεων και ερέθισαν τη μικροστήλη. Με βάση την επικάλυψη οι μικροστήλες συμμετέχουν στον τοπικό ή ολικό ανταγωνισμό μέσω της αναστολής μεταξύ μικροστηλών, ώστε να προκύψουν οι αραιές μικροστήλες που θα ενεργοποιηθούν αυτή τη χρονική στιγμή. Μετά, προσαρμόζονται οι συνάψεις κατά STDP:

- αν η μετασυναπτική μικροστήλη ενεργοποιήθηκε, οι συνάψεις που *μετήγαγαν* ερεθίσματα ισχυροποιούνται
- αν η μετασυναπτική μικροστήλη ενεργοποιήθηκε, οι συνάψεις που *δε μετήγαγαν* ερεθίσματα εξασθενούν
- αν η μετασυναπτική μικροστήλη δεν ενεργοποιήθηκε, οι συνάψεις της δεν προσαρμόζονται

Η καλή λειτουργία του χωρικού συγκεντρωτή (ειδικά η ιδιότητα 5) υποστηρίζεται από ένα σημαντικό επικουρικό μηχανισμό ομοιόστασης που ονομάζεται **«παρώθηση»** (boosting). Η παρώθηση πολλαπλασιάζει την επικάλυψη κάθε μικροστήλης με έναν παράγοντα ενίσχυσης ή εξασθένισης, ανάλογα με το πόσο συχνά ενεργοποιούνταν στο πρόσφατο παρελθόν: ενισχύει τις μικροστήλες που ενεργοποιούνται σπάνια και εξασθενεί τις μικροστήλες που ενεργοποιούνται συχνά. Με αυτόν τον τρόπο επιτυγχάνει πιο ισορροπημένη συμμετοχή όλων των μικροστηλών στη χωρική κωδικοποίηση.

2.3.2 Χρονική Μνήμη

TODO

Κεφάλαιο 3

Υλοποίηση HTM σε Julia

3.1 Υλοποίηση του χωρικού συγκεντρωτή

Στην ενότητα 2.3.1 περιγράφηκε ο αλγόριθμος του χωρικού συγκεντρωτή. Εδώ θα ακολουθήσουμε τη σχεδίαση της υλοποίησής του σε Julia.

3.1.1 Χώροι εισόδου και εξόδου

Ο χωρικός συγκεντρωτής εμπλέκει ένα N_{in}-διάστατο πεπερασμένο διακριτό πλέγμα στην είσοδο με ένα N_{sp}-διάστατο πεπερασμένο διακριτό πλέγμα στην έξοδο, που αντιπροσωπεύουν τα φλοιικά επίπεδα με κάθε πλεγματική κορυφή να αντιστοιχίζεται σε μία μικροστήλη. Το μέγεθος του πλέγματος εισόδου είναι $sz_{in} \in \mathbb{R}^{N_{in}}$ και του πλέγματος εξόδου $sz_{sp} \in \mathbb{R}^{N_{sp}}$. Με βάση αυτό ορίζουμε το μήκος των δύο πλεγμάτων ως $\ell_i = \prod sz_i$

Ας ορίσουμε κάποιες διαστάσεις για να χτίσουμε ένα παράδειγμα:

```
Nᵢₙ= 2
Nₛₚ= 2
szᵢₙ= (5,5)
szₛₚ= (6,9)
```

Ορισμός συνάρτησης που δέχεται σύμβολα ως όρισμα

Ορίσαμε τις διαστάσεις των πλεγμάτων ως σταθερές. Το μήκος των πλεγμάτων όμως δε χρειάζεται να είναι νέες, ανεξάρτητες σταθερές. Μπορεί να είναι συνάρτηση του ονόματος της διάστασης, ώστε, αν αλλάξουμε τη διάσταση, να αλλάξει και το μήκος. Ιδανικά, θα θέλαμε να γράφουμε:

```
julia> ℓ(:in)
64

julia> szᵢₙ= (2,40)
(2,40)

julia> ℓ(:in)
80
```

Ο πιο απλός τρόπος να το καταφέρουμε αυτό είναι ο εξής:

```
julia> ℓ(s::Symbol)= if s === :in
  prod(ᵢₙsz)
elseif s === :sp
  prod(ₛₚsz)
else
  error("Undefined symbol")
endℓ
 (generic function with 1 method)

julia> ℓ(:in)
25

julia> ℓ(:sp)
54

ℓ(:random_symbol)

Error: Undefined symbol
```

Παραπάνω χρησιμοποιήθηκε ο τελεστής της ταύτισης ===, αντί για τον τελεστή της ισότητας ==. Εν προκειμένω θα συμπεριφέρονταν εξίσου, γιατί τα σύμβολα είναι «μοναδικές» (singleton) τιμές.

Ένας εναλλακτικός τρόπος:

```
julia> ℓ(::Val{:in})= prod(szᵢₙ)
ℓ (generic function with 2 methods)

julia> ℓ(::Val{:sp})= prod(szₛₚ)
ℓ (generic function with 3 methods)

julia> ℓ(s::Symbol)= ℓ(Val(s))
ℓ (generic function with 3 methods)

julia> ℓ(:in)
25

julia> ℓ(:sp)
54
```

Ο δεύτερος ορισμός χρησιμοποιεί πιο σύνθετα στοιχεία της γλώσσας: παραμετρικούς τύπους και πολλαπλή αποστολή για να επιλέξει τη σωστή μέθοδο (βλέπε 1.4.1). Ωστόσο, προσφέρει ένα σημαντικό πλεονέκτημα.

Αν θέλουμε να ορίσουμε επιπλέον συμπεριφορά της συνάρτησης για ένα νέο σύμβολο, έστω `:random_symbol`, δε χρειάζεται να ανατρέξουμε στον κώδικα στο σημείο ορισμου της μεθόδου `ℓ(::Symbol)` και να την τροποποιήσουμε ·αρκεί να ορίσουμε μία νέα μέθοδο:

```
julia> ℓ(::Val{:random_symbol})= rand()
ℓ (generic function with 4 methods)

julia> ℓ(:random_symbol)
0.8066650311227195
```

```
julia> ℓ(:random_symbol)
0.36795139800046567
```

Καταδεικνύεται έτσι μια βασική σχεδιαστική αρχή στη Julia που βοηθά στην απόπλεξη του κώδικα: προτιμούμε να ορίζουμε πολλές μικρές μεθόδους και τύπους-περικαλύμματα (wrapping types), όπως εδώ ο `Val`.

Έστω $\mathbb{B} = \{0, 1\}$ το σύνολο των λογικών τιμών, οπότε $x \in \mathbb{B}^N$ είναι ένα δυαδικό διάνυσμα μήκους N.

Έστω τα ερεθίσματα εισόδου $z \in \mathbb{B}^{\ell_{in}}$ και εξόδου $a \in \mathbb{B}^{\ell_{sp}}$. Τα z, a αντιστοιχίζουν τα ερεθίσματα με τις κορυφές του πλέγματος, αν ξεδιπλώσουμε τις διαστάσεις των πλεγμάτων με τη σειρά (column-major).

```
using Random
z= bitrand(ℓ(:in))
```

Αντιστοίχιση χώρων εισόδου-εξόδου

Οι μικροστήλες της εισόδου και της εξόδου αντιστοιχίζονται μεταξύ τους μέσω των εγγύς συνάψεων των νευρώνων τους. Η j-οστή μικροστήλη εισόδου συμβολίζεται x_j κι η i-οστή μικροστήλη εξόδου y_i. Ο χώρος της εισόδου θεωρείται στη γενική περίπτωση ότι έχει τοπολογία, η οποία πρέπει να διατηρηθεί στο χώρο εξόδου. Για αυτό, κάθε y_i αντιστοιχίζεται σε μία περιοχή της εισόδου, έναν υπερκύβο με κέντρο x_i^c και πλευρά γ.

Η αντιστοίχιση $y_i \longleftrightarrow x_i^c$ γίνεται ώστε να κλιμακωθούν οι διαστάσεις των δύο πλεγμάτων και να ταιριάξουν. Ενώ γενικά επιτρέπεται $N_{in} \neq N_{sp}$, δεν υλοποιείται η αντιστοίχιση για τέτοια περίπτωση επειδή δε φαίνεται υψηλής πρακτικής σημασίας.

```
xᶜ(yᵢ)= floor.(Int, (yᵢ.-1) .* (szᵢₙ./szₛₚ)) .+1
```

Οι τελείες `.` στα ονόματα των συναρτήσεων και των τελεστών επιμερίζουν τη συνάρτηση σε κάθε στοιχείο του διανύσματος ή πίνακα. Αυτή είναι μια γενική δυνατότητα της Julia και ονομάζεται «broadcasting».

Ορισμός υπερκύβου

Κάθε μικροστήλη y_i έχει συνάψεις με μερικά από τα $x_j \in \mathit{hypercube}(x_i^c, \gamma)$, επιλεγμένα τυχαία. Καθώς τα πλέγματα στα οποία εργαζόμαστε έχουν σύνορα, οι υπερκύβοι που βρίσκονται κοντά στα σύνορα θα πρέπει να περιέχουν μόνο τα υπαρκτά στοιχεία του πλέγματος, δηλαδή αυτά που βρίσκονται εντός των συνόρων. Πλέγμα μεγέθους `sz` έχει σύνορα στο `{1,sz}`.

Ορίζουμε μία δομή που περιγράφει τον υπερκύβο N διαστάσεων, κέντρου x^c κι ακτίνας γ, εντός πλέγματος μεγέθους `sz`. Από τη δομή απαιτούμε συγκεκριμένη συμπεριφορά: πρέπει να επιτρέπει τον υπολογισμό όλων των κορυφών που ανήκουν στον υπερκύβο. Θα μπορούσαμε βεβαίως να υλοποιήσουμε τη δομή ως έναν πίνακα με όλα τα στοιχεία. Όμως για τόσο απλούς υπολογισμούς είναι πιο συμφέρον να υπολογίζουμε τα στοιχεία επιτόπου. Έτσι, από τη δομή θα απαιτήσουμε να λειτουργεί ως «επαναλήπτης» (iterator): να ορίζει το πρώτο και το τελευταίο στοιχείο και για κάθε στοιχείο να υπολογίζει το επόμενο. Υπάρχει μία δομή στη Julia που επιτελεί έναν κοντινό σκοπό, η `CartesianIndices`. Θα ορίσουμε λοιπόν ένα σύνθετο τύπο που να την περικαλύπτει και να προσφέρει εύκολη κατασκευή και προσπέλαση.

Ο τύπος:

```
struct Hypercube{N}
  xᶜ::NTuple{N,Int}
  γ::Int
  sz::NTuple{N,Int}
  indices::CartesianIndices{N}
end
```

Εύκολη κατασκευή:

```
Hypercube(xᶜ,γ,sz)= Hypercube(xᶜ,γ,sz, start(xᶜ,γ,sz))
start(xᶜ,γ,sz)= CartesianIndices(map( (a,b)-> a:b,
                    max.(xᶜ .- γ, 1),
                    min.(xᶜ .+ γ, sz) ))
```

Προσπέλαση (διεπαφή επαναλήπτη):

```
Base.iterate(hc::Hypercube)= begin
  i= iterate(hc.indices)
  !isnothing(i) ? (i[1].I,i[2]) : nothing
end
Base.iterate(hc::Hypercube,state)= begin
  i= iterate(hc.indices,state)
  !isnothing(i) ? (i[1].I,i[2]) : nothing
end
Base.length(hc::Hypercube)= length(hc.indices)
Base.size(hc::Hypercube)= size(hc.indices)
```

Ας δούμε τι σημεία περιέχει ένας υπερκύβος γύρω από το `(2,2)` με ακτίνα 2 και φραγμένος στο `[1,5]×[1,5]`:

```
julia> using Lazy: @>, @>>

julia> hc= Hypercube((1,2),1,szᵢₙ);

julia> @> hc collect
6—element Array{\{}Any,1{\}}:
 (1, 1)
 (2, 1)
 (1, 2)
 (2, 2)
 (1, 3)
 (2, 3)
```

Για ευκολία στη διατύπωση χρησιμοποιούνται οι μακροεντολές `@>`, `@>>` του πακέτου `Lazy.jl` που επιτρέπουν την «ύφανση» ορισμάτων στις συναρτήσεις [45]:

```
f(g(x))      === @> x g f
f(x,g(x,y)) === @>> y g(x) f(x)
```

Η αναστροφή της σειράς γραφής ενδεχομένως να καθιστά την εφαρμογή των συναρτήσεων πιο ευανάγνωστη.

3.1.2 Κατασκευή εγγύς συνάψεων

Οι εγγύς συνάψεις αναπαρίστανται με έναν πίνακα $W_p \in \mathbb{B}^{\ell_{in} \times \ell_{sp}}$ Η πρώτη διάσταση, της εισόδου, αντιστοιχεί στους προσυναπτικούς νευρώνες κι η δεύτερη, της εξόδου, στους μετασυναπτικούς.

Αυτός ο πίνακας προκύπτει από τον πίνακα $D_p \in \mathbb{S}\mathbb{q}^{\ell_{in} \times \ell_{sp}}$ που περιγράφει τη μονιμότητα κάθε σύναψης:

$$W_{p\cdot} = \begin{cases} 1, & D_{p\cdot} \geq \theta_{conn} \\ 0, & D_{p\cdot} < \theta_{conn} \end{cases} \tag{3.1}$$

όπου $\mathbb{S}\mathbb{q}$ είναι το πεδίο κβάντισης των μονιμοτήτων, που δεν απαιτείται να έχουν συνεχείς τιμές. Στην πράξη, επιλέγουμε $\mathbb{S}\mathbb{q} = \{0x00..0xff\} = UInt8$, θεωρώντας ότι 256 επίπεδα κβάντισης είναι αρκετά για την ομαλή λειτουργία του αλγορίθμου. Αυτό όμως είναι ενδιαφέρον σημείο για περισσότερη μελέτη.

Με δεδομένη την τοπολογία μπορούμε να αρχικοποιήσουμε αυτόν τον πίνακα. Ας ετοιμάσουμε μερικές βοηθητικές συναρτήσεις:

```
const 𝕊𝕢= UInt8
𝕊𝕢range= 𝕊𝕢(0):typemax(𝕊𝕢)
xᵢ(xᶜ)= Hypercube(xᶜ,γ,szᵢₙ)
# Iterate over the output lattice coordinates
out_lattice()= (c.I for c in CartesianIndices(szₛₚ))
θ_effective()= floor(𝕊𝕢, (1 - θ_potential_prob)*typemax(𝕊𝕢))
# Translate cartesian coordinates to linear indices
c2lᵢₙ= LinearIndices(szᵢₙ)
c2lₛₚ= LinearIndices(szₛₚ)

Wₚ()= Dₚ .> θ_permanence
```

Παράμετροι:

```
# θ_potential_prob ∈ [0,1]: κατώφλι που εκφράζει το ποσοστό των εγγύς
   συνάψεων που μπορούν να δημιουργηθούν
# θ_permanence ∈ 𝕊𝕢: κατώφλι σύνδεσης σύναψης
# γ ∈ ℤ: ακτίνα υπερκύβου στο χώρο εισόδου
θ_potential_prob= 0.3
θ_permanence= 0x7f
γ= 2
```

$\forall y_i \forall x_i(y_i)$ θα λάβουμε ένα τυχαίο αριθμό. Αν είναι κάτω από το κατώφλι `θ_effective`, δημιουργείται εγγύς σύναψη $y_i \longleftrightarrow x_i$ με τυχαία μονιμότητα.

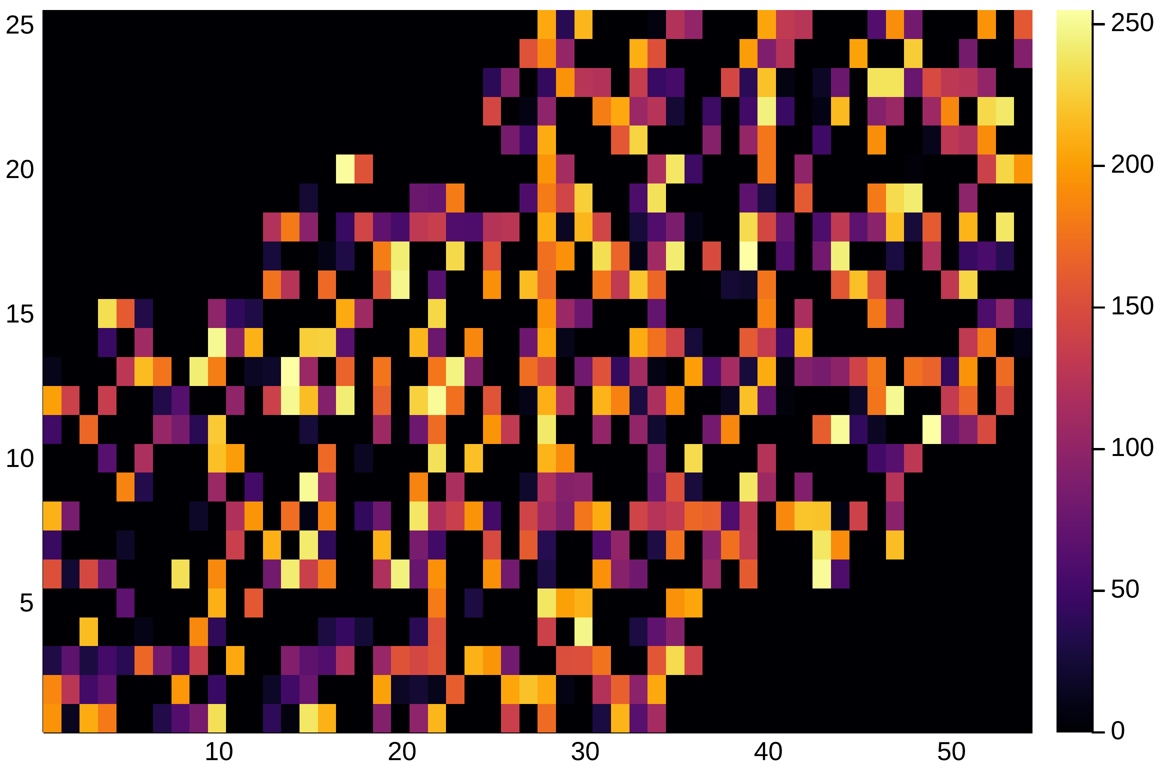
```
permanences(xᵢ)= begin
  # Decide randomly if yᵢ ↔ xᵢ will connect
  p= rand(𝕊𝕢range,length(xᵢ))
  p0= p .> θ_effective(); pScale= p .< θ_effective()
  p[p0].= 𝕊𝕢(0)
  # Draw permanences from uniform distribution in 𝕊𝕢
```

```
  p[pScale].= rand($𝕢range, count(pScale))
  return p
end
fillin_permanences()= begin
  Dₚ= zeros($𝕢, ℓ(:in),ℓ(:sp))
  foreach(out_lattice()) do yᵢ
    # Linear indices from hypercube
    x= @>> yᵢ xᶜ xᵢ collect map(x->c2lᵢₙ[x...])
    Dₚ[x, c2lₛₚ[yᵢ...]]= permanences(@> yᵢ xᶜ xᵢ)
  end
  return Dₚ
end

Dₚ= fillin_permanences()
```

Ο πίνακας D_p που μόλις ορίσαμε είναι **η μόνη μεταβλητή κατάστασης** του χωρικού συγκεντρωτή ως τώρα. Όλοι οι άλλοι κεντρικοί ορισμοί είναι συναρτήσεις, δηλαδή δεν ορίζουν κατάσταση.

Ας οπτικοποιήσουμε τις συνάψεις:

```
using Plots; gr()
heatmap(Dₚ)
```



3.1.3 Ενεργοποίηση Χωρικού Συγκεντρωτή

Στον υπολογισμό της ενεργοποίησης υπάρχουν 3 φάσεις:

- Επικάλυψη $o(y_i)$, δηλαδή άθροιση ερεθισμάτων για κάθε y_i μέσω συνδεδεμένων συνάψεων
- Εφαρμογή παρώθησης b ανά y_i
- Αναστολή μεταξύ y_i ανά γειτονιά
- Ενεργοποίηση μικροστηλών που κέρδισαν την αναστολή και έχουν ερεθιστεί > από ένα κατώφλι `θ_stimulus_activate`

```
θ_stimulus_activate= 1
```

Επικάλυψη

Η παρώθηση είναι συμπληρωματική διαδικασία, οπότε αρχικά την παραλείπουμε. Η επικάλυψη `o` για κάθε y_i ως προς την είσοδο `z` είναι απλώς ένας **πολλαπλασιασμός πίνακα × διάνυσμα**.

```
o(z)= @> Wₚ()'z reshape(szₛₚ)
```

Τοπική αναστολή

Η αναστολή εφαρμόζεται σε κυλιόμενο παράθυρο του πλέγματος y ακτίνας φ, που ονομάζεται ακτίνα αναστολής. Για κάθε παράθυρο, επιλέγουμε τα ky_i με τη μεγαλύτερη επικάλυψη. Έστω $Z(X,p)$ το p-εκατοστημόριο του διανύσματος X. Η αναστολή παραμετροποιείται από το «ποσοστό τοπικής αραιότητας» s (συνήθως 2%). Οπότε $k \approx \textit{ceil}(s\textit{area})$, όπου $\textit{area} = (2\varphi+1)^{N_{sp}}$ ο αριθμός κορυφών του πλέγματος y ανά γειτονιά.

Ένας καλός τρόπος για να υπολογίσουμε το $Z(X,(1-s))$ είναι η μερική ταξινόμηση των επικαλύψεων της γειτονιάς με τον αλγόριθμο «quickselect»:

```
z!(X)= @> X vec partialsort!(k(),rev=true)
```

Τώρα η επικάλυψη ανά περιοχή γίνεται:

```
import ImageFiltering: mapwindow, Fill
a(φ)= 2round(Int,φ)+1
k()= ceil(Int, s*a(φ)^Nₛₚ)
Z(o)= mapwindow(z!, o, window(), border= Fill(0))
window()= ntuple(i->a(φ),Nₛₚ)
```

Η σχέση της τοπικής αραιότητας s και της αραιότητας ολόκληρου του επιπέδου (`sparsity(a)= count(a)/length(a)`) δεν είναι προφανής. Για μεγάλο s και γ τείνει `s ≈ sparsity(a)`. Όμως για μικρά `s` και γ, επειδή πρέπει $k \in \mathbb{Z}$, μπορεί το `k` να μην αντιπροσωπεύει ακριβώς το `s`. Επίσης για μικρό μέγεθος επιπέδων είναι πιθανό πολλά y_i να έχουν το ίδιο $o \in \mathbb{Z}$. Αν ενεργοποιηθούν όλα τα $y_i | o(y_i) \geq Z(o,(1-s))$, ή ακόμα και μόνο αυτά που $o(y_i) > Z(o,(1-s))$, ενδέχεται να ενεργοποιηθούν αντίστοιχα περισσότερα ή λιγότερα y_i από k. Θα μπορούσαμε βέβαια να ενεργοποιήσουμε ακριβώς k κρατώντας την πρώτη ενεργοποίηση και επιλύοντας τυχαία τις ισοπαλίες. Πρακτικά αυτή η παρατήρηση δε φαίνεται να επηρεάζει τη λειτουργία του χωρικού συγκεντρωτή, οπότε επιλέγεται η απλούστερη λύση χωρίς τυχαία επίλυση των ισοπαλιών.

Η ακτίνα αναστολής φ θεωρητικά είναι δυναμική και αυξάνει καθώς αυξάνεται το μέσο υποδεκτικό πεδίο των μικροστηλών εξόδου. Όμως στην πράξη η ρύθμιση αυτή φαίνεται να είναι αμελητέα και σε πρώτη ανάλυση αγνοείται. Ας προσδιορίσουμε λοιπόν την αρχική (και με την προηγούμενη λογική, μόνιμη) τιμή της φ ως προς την τιμή της αντίστοιχης παραμέτρου γ του χώρου εισόδου:

```
using StatsBase: mean, median
φ= γ*mean(szₛₚ./szᵢₙ)
s= 0.1
```

Με βάση τα παραπάνω, η ενεργοποίηση των μικροστηλών εξόδου είναι:

```
activate(o)= ((o .>= Z(o)) .& (o .> θ_stimulus_activate))|> vec
```

Άρα η ενεργοποίηση ως προς την είσοδο `z`:

```
julia> reshape(z,szᵢₙ)
5×5 BitArray{\{}2{\}}:
  true   true   true   true   true
  true  false   true  false   true
  true  false  false   true  false
  true  false  false  false  false
 false   true   true   true   true

julia> a= z|> o|> activate;

julia> reshape(a,szₛₚ)
6×9 BitArray{\{}2{\}}:
  true  false  false  false   true   true  false  false  false
 false  false  false  false  false   true  false  false   true
  true  false  false  false  false  false  false  false  false
 false   true  false   true   true  false   true  false   true
 false  false  false  false  false  false  false  false   true
 false  false  false  false   true  false  false  false  false
```

Παρώθηση

Ο χωρικός συγκεντρωτής χρησιμοποιεί όπως αναφέρθηκε στην ενότητα 2.3.1 ένα μηχανισμό που αυξάνει την εντροπία της εξόδου, ευαισθητοποιώντας τα y_i που ενεργοποιούνται σπάνια. Ο μηχανισμός υλοποιείται με ένα διάνυσμα b που πολλαπλασιάζει την επικάλυψη των y_i πριν την εφαρμογή της αναστολής:

```
o(z)= @> (b() .* Wₚ()'z) reshape(szₛₚ)
```

Το διάνυσμα αυτό προκύπτει από τη μέση δραστηριότητα κάθε y_i στο χρόνο $\mathring{a}_t$ και τη μέση δραστηριότητα στη γειτονιά κάθε y_i, $\mathring{a}_n$. Εισάγεται έτσι μια νέα μεταβλητή κατάστασης, $\mathring{a}_t$ κι η συνάρτηση μετάβασης/βήματός της. Για τη συνάρτηση βήματος θα χρησιμοποιηθεί η ίδια λογική με παραμέτρους τύπου που εφαρμόστηκε για το ℓ (3.1.1). Το $\mathring{a}_t$ ανανεώνεται ως εκθετικό φίλτρο κυλιόμενου μέσου όρου με περίοδο `Tboost`. Το $\mathring{a}_n$ είναι συνάρτηση του $\mathring{a}_t$: κυλιόμενο φίλτρο εικόνας με πυρήνα μέσου όρου.

```
using ImageFiltering: imfilter
# Exponential Moving Average
```

```
step!(::Val{:åₜ}, åₜ,a)= ( åₜ.= (åₜ.*(Tboost-1) .+
   reshape(a,szₛₚ))./Tboost )
# Mean filtering of an image
mean_kernel()= ones(window()) ./ α(φ).^Nₛₚ
åₙ()= imfilter(åₜ, mean_kernel(), "symmetric")
# a convenient wrapper
step!(s::Symbol, args...)= step!(Val(s), args...)
```

Το b είναι ως προς αυτά:

```
b()= @> exp.(-β .* (åₜ.-åₙ())) vec
```

Απαιτούνται δύο νέες παράμετροι, η περίοδος του φίλτρου κυλιόμενου μέσου και η ισχύς της παρώθησης:

```
Tboost= 200
β= 1
```

Αρχικοποιώντας το åₜ με τυχαίες τιμές για να δούμε το αποτέλεσμα, μπορούμε να λάβουμε τη νέα ενεργοποίηση της εξόδου

```
julia> åₜ= rand(szₛₚ...);

julia> reshape(z,szᵢₙ)
5×5 BitArray{\{}2{\}}:
  true   true   true   true   true
  true  false   true  false   true
  true  false  false   true  false
  true  false  false  false  false
 false   true   true   true   true

julia> reshape(a,szₛₚ)
6×9 BitArray{\{}2{\}}:
  true  false  false  false   true   true  false  false  false
 false  false  false  false  false   true  false  false   true
  true  false  false  false  false  false  false  false  false
 false   true  false   true   true  false   true  false   true
 false  false  false  false  false  false  false  false   true
 false  false  false  false   true  false  false  false  false

julia> a_boosted= z|> o|> activate;

julia> reshape(a_boosted,szₛₚ)
6×9 BitArray{\{}2{\}}:
  true  false  false  false  false   true  false  false  false
 false  false  false  false  false   true  false  false  false
 false  false  false  false  false  false  false  false  false
 false  false  false   true   true  false  false  false   true
 false  false  false  false  false  false  false  false   true
 false  false  false  false   true  false  false  false  false
```

και να διαπιστώσουμε πώς γίνεται ως τώρα η ενημέρωση της κατάστασης.

```
julia> reshape(åₜ,szₛₚ)
```

```
6×9 Array{\{}Float64,2{\}}:
 0.237717   0.166215  0.845729  0.489799   …  0.713044  0.325335
   0.860011
 0.0414733  0.960093  0.741242  0.546676      0.994453  0.14325
   0.692403
 0.90726    0.313701  0.583062  0.0523834     0.680307  0.570053
   0.137405
 0.835714   0.672971  0.113465  0.174871      0.834358  0.679982
   0.32576
 0.295988   0.961276  0.718726  0.745415      0.763077  0.358674
   0.127556
 0.504287   0.837043  0.751036  0.813725   …  0.331168  0.599893
   0.462521

julia> step!(:åₜ, åₜ,a_boosted);

julia> reshape(åₜ,szₛₚ)
6×9 Array{\{}Float64,2{\}}:
 0.241528   0.165384  0.8415    0.48735    …  0.709479  0.323708
   0.855711
 0.0412659  0.955293  0.737535  0.543942      0.989481  0.142534
   0.688941
 0.902724   0.312132  0.580147  0.0521215     0.676906  0.567202
   0.136718
 0.831535   0.669606  0.112898  0.178997      0.830186  0.676582
   0.329131
 0.294508   0.95647   0.715132  0.741688      0.759262  0.35688
   0.131918
 0.501766   0.832857  0.74728   0.809657   …  0.329512  0.596893
   0.460208

julia> @> z o activate reshape(szₛₚ)
6×9 BitArray{\{}2{\}}:
  true  false  false  false  false   true  false  false  false
 false  false  false  false  false   true  false  false  false
 false  false  false  false  false  false  false  false  false
 false  false  false   true   true  false  false  false   true
 false  false  false  false  false  false  false  false   true
 false  false  false  false   true  false  false  false  false
```

Είναι σημαντικό να λογαριάζουμε τις μεταβλητές κατάστασης του συστήματος, που ως τώρα είναι 2, `Dₚ` και `åₜ`, γιατί καθεμία χρειάζεται εφαρμογή συνάρτησης μετάβασης σε κάθε χρονικά βήμα του χωρικού συγκεντρωτή. Στην επόμενη ενότητα θα ορίσουμε τη συνάρτηση μετάβασης των συνάψεων (εκμάθηση) και θα συγκεντρώσουμε όλες τις συναρτήσεις μετάβασης σε μία, `step!(::Val{:sp})`.

3.1.4 Εκμάθηση (προσαρμογή συνάψεων)

Οι συνάψεις των ενεργοποιημένων y_i προσαρμόζονται, ενισχύοντας τις συνάψεις προς τα ενεργά x_i και εξασθενώντας τις συνάψεις με τα ανενεργά. Σε αυτόν τον κανόνα όμως εμπίπτουν μόνο οι συνάψεις που απέκτησαν αρχική τιμή 0. Πολλές θέσεις του πίνακα `Dₚ[i,j] == 0` συμβολίζουν έλλειψη σύναψης [1] και δεν πρέπει να αλλάξουν.

Ο απλούστερος τρόπος για να το επιτύχουμε είναι ο εξής:

```
learn!(Dₚ,z,a)= begin
  Dₚ[z,a]  .= (Dₚ[z,a].>0) .* (Dₚ[z,a]   .⊕ p⁺)
  Dₚ[.!z,a].= (Dₚ[z,a].>0) .* (Dₚ[.!z,a] .⊖ p⁻)
end
```

Οι επιπλέον παράμετροι `p⁺`, `p⁻` εκφράζουν το πόσο αυξομειώνονται οι συνάψεις σε κάθε βήμα εκμάθησης.

```
p⁺= 0.1; p⁻= 0.02
p⁺= round($𝕢, p⁺*typemax($𝕢)); p⁻= round($𝕢, p⁻*typemax($𝕢))
```

Οι τελεστές `⊕`, `⊖` που εφαρμόζουν τη συναπτική τροποποίηση είναι κορεσμένη πρόσθεση και αφαίρεση αντιστοίχως.

```
⊕(a::T,b::T) where {T<:Unsigned}= a+b>a ? a+b : typemax(T)
⊖(a::T,b::T) where {T<:Unsigned}= a-b<a ? a-b : one(T)
⊕(a::T,b::T) where {T<:Signed}= a+b>a ? a+b : typemax(T)
⊖(a::T,b::T) where {T<:Signed}= a-b>0 ? a-b : one(T)
```

Βελτίωση

Βέβαια η παραπάνω μέθοδος δεν είναι πολύ αποδοτική για δύο λόγους:

- προσπελαύνει πολλές φορές έναν εν δυνάμει πολύ μεγάλο πίνακα
- δημιουργεί αντίγραφά του

Το τελευταίο οφείλεται στον τρόπο με τον οποίο λειτουργεί η λήψη στοιχείων από πίνακα στη Julia. `A[i,j]` σημαίνει δημιουργία ενός νέου πίνακα, μεγέθους `ℓ(i)×ℓ(j)`, με τα στοιχεία που περιέχει ο `A` στην αντίστοιχη περιοχή.

Για να αποφύγουμε αυτή τη συμπεριφορά μπορούμε να χρησιμοποιήσουμε «όψεις» (array views). Μια βελτιωμένη εκδοχή λοιπόν είναι:

```
learn!(Dₚ,z,a)= begin
  Dₚactive= @view Dₚ[:,a]   # the only elements we touch
  activeConn=   (Dₚactive .> 0) .&   z
  inactiveConn= (Dₚactive .> 0) .& .!z
  Dₚactive.= activeConn   .* (Dₚactive .⊕ p⁺) .+
             inactiveConn .* (Dₚactive .⊖ p⁻)
end
step!(::Val{:Dₚ}, Dₚ,z,a)= learn!(Dₚ,z,a)
```

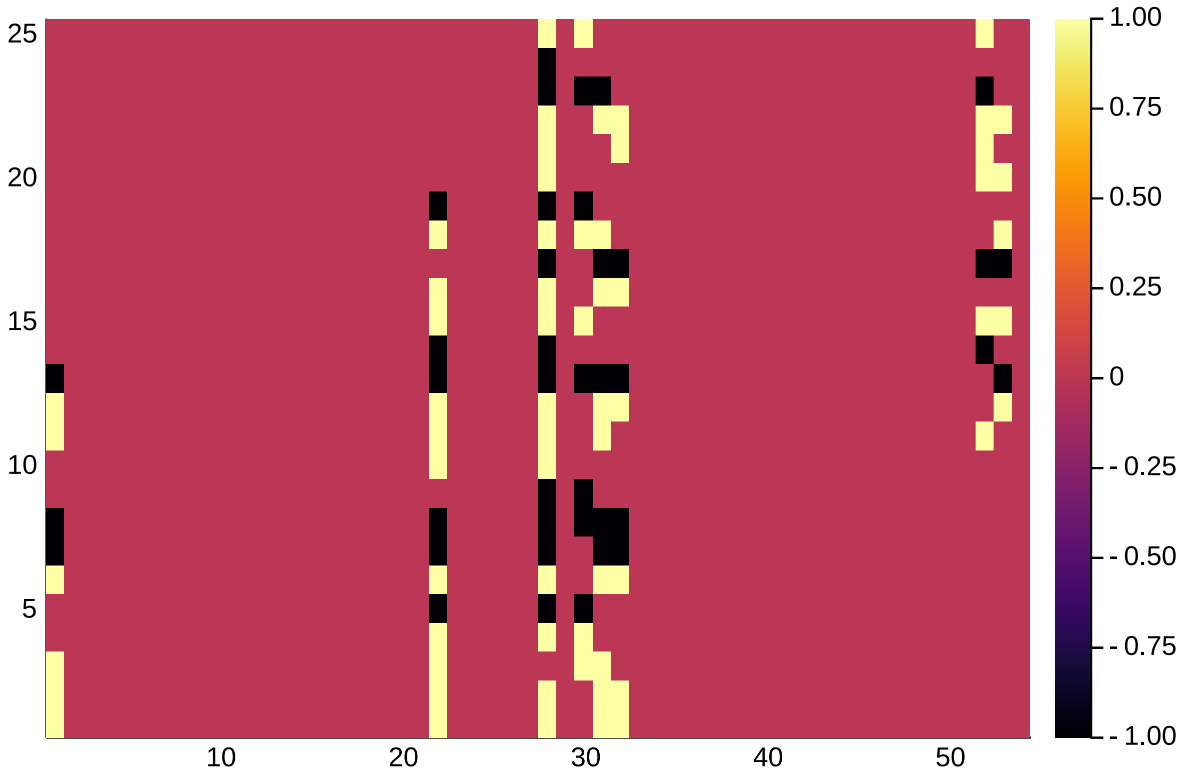
Ας δούμε πώς τροποποιούνται οι συνάψεις σε έναν κύκλο ενεργοποίησης του χωρικού συγκεντρωτή.

[1] Τα κενά δεν είναι επαρκή εν προκειμένω για να θεωρήσουμε τον D_p αραιό

```
julia> a= z|> o|> activate;

julia> reshape(a,szₛₚ)
6×9 BitArray{\{}2{\}}:
  true  false  false  false  false   true  false  false  false
 false  false  false  false  false   true  false  false  false
 false  false  false  false  false  false  false  false  false
 false  false  false   true   true  false  false  false   true
 false  false  false  false  false  false  false  false   true
 false  false  false  false   true  false  false  false  false

julia> D_before= copy(Dₚ);

julia> step!(:Dₚ, Dₚ,z,a);

julia> step!(:åₜ, åₜ,a);

julia> a= z|> o|> activate;

julia> reshape(a,szₛₚ)
6×9 BitArray{\{}2{\}}:
  true  false  false  false  false   true  false  false  false
 false  false  false  false  false   true  false  false  false
 false  false  false  false  false  false  false  false  false
 false  false   true   true   true  false  false  false   true
 false  false  false  false  false  false  false  false   true
 false  false  false  false   true  false  false  false  false
```

Στις ενεργές μικροστήλες εξόδου, κάποιες συνάψεις ενισχύονται και κάποιες εξασθενούν:

```
D_diff= Int.(Dₚ .> D_before) - Int.(Dₚ .< D_before);
D_diff|> heatmap
```



3.2 Σύνθεση στοιχείων χωρικού συγκεντρωτή

Το μόνο εμπόδιο που μένει για τη χρήση του χωρικού συγκεντρωτή είναι ότι οι προηγούμενοι ορισμοί χρησιμοποιούσαν «global» μεταβλητές για την κατάσταση και για τις παραμέτρους. Αυτό βοήθησε να απλοποιήσουμε την περιγραφή του αλγορίθμου, αλλά όχι τη χρήση του ως βιβλιοθήκη. Καθώς ο έλεγχος της κατάστασης είναι κρίσιμος για τη συμπερασματολογία σε οποιοδήποτε πρόγραμμα, θα πρέπει όλες οι «global» μεταβλητές να εξαλειφθούν. Οι τρόποι αντιμετώπισης είναι:

- να περικλείσουμε τον ορισμό των περισσότερων εξειδικευμένων συναρτήσεων εντός του πεδίου γενικότερων συναρτήσεων, όπου θα έχουν έμμεση πρόσβαση στα σύμβολα που ορίζουν οι περικλείουσες
- να δώσουμε στις γενικότερες συναρτήσεις που μενουν επιπλέον ορίσματα

Οι ορισμοί που δίνονται παρακάτω συγκεντρώνουν όλη την υλοποίηση του χωρικού συγκεντρωτή και στέκουν αυτόνομα από τους προηγούμενους. Δε γίνεται καμία αλλαγή στους αλγορίθμους.

Ολόκληρος ο χωρικός συγκεντρωτής υλοποιείται σε **140 γραμμές κώδικα**.

Ορισμοί σταθερών και τύπων υποδομής:

```
using Random
using Parameters
import Lazy: @>, @>>
import StatsBase: mean, median
import ImageFiltering: mapwindow, Fill, imfilter
```

```julia
# Constants
const 𝕊𝕢= UInt8
const 𝕊𝕢range= 𝕊𝕢(0):typemax(𝕊𝕢)
# Topology
struct Hypercube{N}
  xᶜ::NTuple{N,Int}
  γ::Int
  sz::NTuple{N,Int}
  indices::CartesianIndices{N}
end
Hypercube(xᶜ,γ,sz)= Hypercube(xᶜ,γ,sz, start(xᶜ,γ,sz))
start(xᶜ,γ,sz)= CartesianIndices(map( (a,b)-> a:b,
                    max.(xᶜ .- γ, 1),
                    min.(xᶜ .+ γ, sz) ))
Base.iterate(hc::Hypercube)= begin
  i= iterate(hc.indices)
  !isnothing(i) ? (i[1].I,i[2]) : nothing
end
Base.iterate(hc::Hypercube,state)= begin
  i= iterate(hc.indices,state)
  !isnothing(i) ? (i[1].I,i[2]) : nothing
end
Base.length(hc::Hypercube)= length(hc.indices)
Base.size(hc::Hypercube)= size(hc.indices)
# Infrastructure
ℓ(sz)= prod(sz)
step!(s::Symbol,args...)= step!(Val(s),args...)
```

Ας συνθέσουμε την αρχικοποίηση της κατάστασης και τις παραμέτρους. Το πακέτο `Parameters` μας διευκολύνει στον ορισμό των παραμέτρων, επιτρέποντας να δώσουμε αρχικές τιμές και βασικό έλεγχο ορθότητας με απλή σύνταξη.

```julia
@with_kw struct SPParams{Nin,Nsp}
  szᵢₙ::NTuple{Nin,Int}     = (32,32); @assert all(szᵢₙ.>0)
  szₛₚ::NTuple{Nsp,Int}     = (64,64); @assert all(szₛₚ.>0)
  γ::Int                    = 6;       @assert γ>0
  φ::Float32                = γ*mean(szₛₚ./szᵢₙ)
  s::Float32                = .02;     @assert s>0
  θ_potential_prob::Float32= .5;       @assert 0<=θ_potential_prob<=1
  θ_permanence01            = .5;      @assert 0<=θ_permanence01<=1
  p⁺_01                     = .1;      @assert 0<=p⁺_01<=1
  p⁻_01                     = .02;     @assert 0<=p⁻_01<=1
  θ_permanence::𝕊𝕢        = @>> θ_permanence01*typemax(𝕊𝕢) round(𝕊𝕢)
  p⁺::𝕊𝕢                  = round(𝕊𝕢, p⁺_01*typemax(𝕊𝕢))
  p⁻::𝕊𝕢                  = round(𝕊𝕢, p⁻_01*typemax(𝕊𝕢))
  θ_stimulus_activate::Int = 1;        @assert θ_stimulus_activate>=0
  Tboost::Float32           = 200;     @assert Tboost>0
  β::Float32                = 1;       @assert β>0
  @assert φ>=1
  @assert zero(𝕊𝕢)<=θ_permanence<=typemax(𝕊𝕢)
  @assert zero(𝕊𝕢)<=p⁺<=typemax(𝕊𝕢)
  @assert zero(𝕊𝕢)<=p⁻<=typemax(𝕊𝕢)
```

```
end
struct SpatialPooler{Nin,Nsp}
  params::SPParams{Nin,Nsp}
  Dₚ::Matrix{Sq}
  åₜ::Array{Float32,Nsp}

  SpatialPooler(params::SPParams{Nin,Nsp}) where {Nin,Nsp}= begin
    @unpack szᵢₙ,szₛₚ,θ_potential_prob,γ = params

    xᶜ(yᵢ)= floor.(Int, (yᵢ.-1) .* (szᵢₙ./szₛₚ)) .+1
    xᵢ(xᶜ)= Hypercube(xᶜ,γ,szᵢₙ)
    θ_effective()= floor(Sq, (1 - θ_potential_prob)*typemax(Sq))
    out_lattice()= (c.I for c in CartesianIndices(szₛₚ))
    permanences(xᵢ)= begin
      # Decide randomly if yᵢ ↔ xᵢ will connect
      p= rand(Sqrange,length(xᵢ))
      p0= p .> θ_effective(); pScale= p .< θ_effective()
      p[p0].= Sq(0)
      # Draw permanences from uniform distribution in Sq
      p[pScale].= rand(Sqrange, count(pScale))
      return p
    end
    fillin_permanences()= begin
      Dₚ= zeros(Sq, ℓ(szᵢₙ),ℓ(szₛₚ))
      foreach(out_lattice()) do yᵢ
        # Linear indices from hypercube
        x= @>> yᵢ xᶜ xᵢ collect map(x->c2lᵢₙ[x...])
        Dₚ[x, c2lₛₚ[yᵢ...]]= permanences(@> yᵢ xᶜ xᵢ)
      end
      return Dₚ
    end
    c2lᵢₙ= LinearIndices(szᵢₙ)
    c2lₛₚ= LinearIndices(szₛₚ)

    new{Nin,Nsp}(params,
                 fillin_permanences(),
                 zeros(szₛₚ...))
  end
end
```

Η συνάρτηση ενεργοποίησης γίνεται:

```
sp_activate(z,sp::SpatialPooler{Nin,Nsp}) where {Nin,Nsp}= begin
  @unpack szₛₚ,s,φ,β,θ_permanence,θ_stimulus_activate = sp.params
  @unpack Dₚ,åₜ = sp
  # overlap
  Wₚ()=  Dₚ .> θ_permanence
  o(z)=  @> (b() .* Wₚ()'z) reshape(szₛₚ)
  # inhibition
  α(φ)=  2round(Int,φ)+1
  k()=   ceil(Int, s*α(φ)^Nsp)
  z!(X)= @> X vec partialsort!(k(),rev=true)
```

```
  Z(o)=  mapwindow(z!, o, window(), border= Fill(0))
  # boosting
  window()= ntuple(i->α(φ),Nsp)
  mean_kernel()= ones(window()) ./ α(φ).^Nsp
  åₙ()=  imfilter(åₜ, mean_kernel(), "symmetric")
  b()=   @> exp.(-β .* (åₜ .- åₙ())) vec

  activate(o)= ((o .>= Z(o)) .& (o .> θ_stimulus_activate))|> vec
  z|> o|> activate
end
```

Ας συνθέσουμε τη συνάρτηση βήματος του χωρικού συγκεντρωτή. Η συνάρτηση λαμβάνει την είσοδο, υπολογίζει την ενεργοποίηση και πραγματοποιεί τις απαιτούμενες μεταβολές κατάστασης.

```
step!(::Val{:sp}, sp,z)= begin
  a= sp_activate(z,sp)
  step!(:åₜ, sp,a)
  step!(:Dₚ, sp,z,a)
  return a
end
step!(::Val{:åₜ}, sp,a)= begin
  @unpack Tboost, szₛₚ = sp.params
  sp.åₜ.= (sp.åₜ.*(Tboost-1) .+ reshape(a,szₛₚ))./Tboost
end
step!(::Val{:Dₚ}, sp,z,a)= learn!(sp.Dₚ,z,a,sp.params)
learn!(Dₚ,z,a,params)= begin
  @unpack p⁺,p⁻ = params
  Dₚactive= @view Dₚ[:,a]   # the only elements we touch
  activeConn=   (Dₚactive .> 0) .&   z
  inactiveConn= (Dₚactive .> 0) .& .!z
  Dₚactive.= activeConn   .* (Dₚactive .⊕ p⁺) .+
             inactiveConn .* (Dₚactive .⊖ p⁻)
end
# saturated arithmetic
⊕(a::T,b::T) where {T<:Unsigned}= a+b>a ? a+b : typemax(T)
⊖(a::T,b::T) where {T<:Unsigned}= a-b<a ? a-b : one(T)
⊕(a::T,b::T) where {T<:Signed}= a+b>a ? a+b : typemax(T)
⊖(a::T,b::T) where {T<:Signed}= a-b>0 ? a-b : one(T)
```

Κεφάλαιο 4

Ανοιχτές ερωτήσεις

Βιβλιογραφία

[1] M. Mohammadi, A. Al-Fuqaha, S. Sorour και M. Guizani, «Deep Learning for IoT Big Data and Streaming Analytics: A Survey,» **IEEE Communications Surveys Tutorials**, τόμ. 20, αρθμ. 4, σσ. 2923–2960, Fourthquarter 2018, ISSN: 1553-877X. DOI: 10.1109/COMST.2018.2844341.

[2] J. Piaget, **The Origins of Intelligence in Children**, σύνταξη υπό M. Cook, σειρά The Origins of Intelligence in Children. New York, NY, US: W W Norton & Co, 1952, 419 **pagetotals**. DOI: 10.1037/11494-000.

[3] P. W. Anderson, «More Is Different,» **Science**, τόμ. 177, αρθμ. 4047, σσ. 393–396, 4 Αύγ. 1972, ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.177.4047.393. pmid: 17796623.

[4] M. Minsky, **Society Of Mind**. Simon and Schuster, 15 Μαρ. 1988, 342 **pagetotals**, ISBN: 978-0-671-65713-0.

[5] D. B. Lenat και E. A. Feigenbaum, «On the Thresholds of Knowledge,» **Artificial Intelligence**, τόμ. 47, αρθμ. 1, σσ. 185–250, 1 Ιαν. 1991, ISSN: 0004-3702. DOI: 10.1016/0004-3702(91)90055-O.

[6] P. Kanerva, «Associative Neural Memories,» στο, M. H. Hassoun, επιμελητής, New York, NY, USA: Oxford University Press, Inc., 1993, κεφ. Sparse Distributed Memory and Related Models, σσ. 50–76, ISBN: 0-19-507682-6.

[7] P. Wang, «On the Working Definition of Intelligence,» 1995.

[8] V. B. Mountcastle, «The columnar organization of the neocortex,» **Brain: A Journal of Neurology**, τόμ. 120 (Pt 4), σσ. 701–722, Απρ. 1997, ISSN: 0006-8950. DOI: 10.1093/brain/120.4.701. pmid: 9153131.

[9] K. Doya, «What are the computations of the cerebellum, the basal ganglia and the cerebral cortex?» **Neural Networks: The Official Journal of the International Neural Network Society**, τόμ. 12, αρθμ. 7-8, σσ. 961–974, Οκτ. 1999, ISSN: 1879-2782. pmid: 12662639.

[10] S. Song, K. D. Miller και L. F. Abbott, «Competitive Hebbian learning through spike-timing-dependent synaptic plasticity,» **Nature Neuroscience**, τόμ. 3, αρθμ. 9, σ. 919, Σεπτ. 2000, ISSN: 1546-1726. DOI: 10.1038/78829.

[11] J. C. Horton και D. L. Adams, «The Cortical Column: A Structure without a Function,» **Philosophical Transactions of the Royal Society B: Biological Sciences**, τόμ. 360, αρθμ. 1456, σσ. 837–862, 29 Απρ. 2005, ISSN: 0962-8436. DOI: 10.1098/rstb.2005.1623. pmid: 15937015.

[12] S. Legg και M. Hutter, «A Collection of Definitions of Intelligence,» στο **Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms**, IOS Press, 2007, σσ. 17–24.

[13] T. Freund και S. Kali, «Interneurons,» **Scholarpedia**, τόμ. 3, αρθμ. 9, σ. 4720, 1 Σεπτ. 2008, ISSN: 1941-6016. DOI: 10.4249/scholarpedia.4720.

[14] P. Wang, «What Do You Mean by “AI”?,» **presentedat**Proceedings of the 2008 Conference on Artificial General Intelligence 2008: Proceedings of the First AGI Conference, IOS Press, 20 Ιούν. 2008, σσ. 362–373, ISBN: 978-1-58603-833-5. διεύθν.: http://dl.acm.org/citation.cfm?id=1566174.1566207 (επίσκεψη 28/05/2019).

[15] J. Defelipe, H. Markram και K. S. Rockland, «The Neocortical Column,» **Frontiers in Neuroanatomy**, τόμ. 6, 2012, ISSN: 1662-5129. DOI: 10.3389/fnana.2012.00022.

[16] E. R. Kandel, T. M. Jessell, J. H. Schwartz, S. A. Siegelbaum και A. J. Hudspeth, **Principles of Neural Science, Fifth Edition**. McGraw Hill Professional, 2013, 1761 **pagetotals**, ISBN: 978-0-07-139011-8.

[17] R. Paton, H. Bolouri, W. M. L. Holcombe, J. H. Parish και R. Tateson, **Computation in Cells and Tissues: Perspectives and Tools of Thought**. Springer Science & Business Media, 14 Μαρ. 2013, 349 **pagetotals**, ISBN: 978-3-662-06369-9.

[18] M. Benedek, E. Jauk, M. Sommer, M. Arendasy και A. C. Neubauer, «Intelligence, Creativity, and Cognitive Control: The Common and Differential Involvement of Executive Functions in Intelligence and Creativity,» **Intelligence**, τόμ. 46, σσ. 73–83, 1 Σεπτ. 2014, ISSN: 0160-2896. DOI: 10.1016/j.intell.2014.05.007.

[19] Y. Bengio, D.-H. Lee, J. Bornschein, T. Mesnard και Z. Lin, «Towards Biologically Plausible Deep Learning,» 13 Φεβ. 2015. arXiv: 1502.04156 [cs]. διεύθν.: http://arxiv.org/abs/1502.04156 (επίσκεψη 28/05/2019).

[20] S. Billaudelle και S. Ahmad, «Porting HTM Models to the Heidelberg Neuromorphic Computing Platform,» 8 Μάι. 2015. arXiv: 1505.02142 [cs, q-bio]. διεύθν.: http://arxiv.org/abs/1505.02142 (επίσκεψη 02/06/2019).

[21] F. De Sousa Webber, «Semantic Folding Theory And its Application in Semantic Fingerprinting,» **ArXiv e-prints**, Νοέ. 2015. arXiv: 1511.08855 [cs.AI].

[22] H. Markram, E. Muller, S. Ramaswamy, M. W. Reimann, M. Abdellah, C. A. Sanchez, A. Ailamaki, L. Alonso-Nanclares, N. Antille, S. Arsever, G. A. A. Kahou, T. K. Berger, A. Bilgili, N. Buncic, A. Chalimourda, G. Chindemi, J.-D. Courcol, F. Delalondre, V. Delattre, S. Druckmann, R. Dumusc, J. Dynes, S. Eilemann, E. Gal, M. E. Gevaert, J.-P. Ghobril, A. Gidon, J. W. Graham, A. Gupta, V. Haenel, E. Hay, T. Heinis, J. B. Hernando, M. Hines, L. Kanari, D. Keller, J. Kenyon, G. Khazen, Y. Kim, J. G. King, Z. Kisvarday, P. Kumbhar, S. Lasserre, J.-V. Le Bé, B. R. C. Magalhães, A. Merchán-Pérez, J. Meystre, B. R. Morrice, J. Muller, A. Muñoz-Céspedes, S. Muralidhar, K. Muthurasa, D. Nachbaur, T. H. Newton, M. Nolte, A. Ovcharenko, J. Palacios, L. Pastor, R. Perin, R. Ranjan, I. Riachi, J.-R. Rodríguez, J. L. Riquelme, C. Rössert, K. Sfyrakis, Y. Shi, J. C. Shillcock, G. Silberberg, R. Silva, F. Tauheed, M. Telefont, M. Toledo-Rodriguez, T. Tränkler, W. Van Geit, J. V. Díaz, R. Walker, Y. Wang, S. M. Zaninetta, J. DeFelipe, S. L. Hill, I. Segev και F. Schürmann, «Reconstruction and Simulation of Neocortical Microcircuitry,» **Cell**, τόμ. 163, αρθμ. 2, σσ. 456–492, 8 Οκτ. 2015, ISSN: 1097-4172. DOI: 10.1016/j.cell.2015.09.029. pmid: 26451489.

[23] K. Meier, «A Mixed-Signal Universal Neuromorphic Computing System,» στο **2015 IEEE International Electron Devices Meeting (IEDM)**, Δεκ. 2015, σσ. 4.6.1–4.6.4. DOI: 10.1109/IEDM.2015.7409627.

[24] A. Adamatzky, επιμελητής, **Advances in Physarum Machines: Sensing and Computing with Slime Mould**, Emergence, Complexity and Computation, Springer International Publishing, 2016, ISBN: 978-3-319-26661-9. διεύθν.: https://www.springer.com/gp/book/9783319266619 (επίσκεψη 29/05/2019).

[25] Y. Cui, S. Ahmad και J. Hawkins, «Continuous Online Sequence Learning with an Unsupervised Neural Network Model,» **Neural Computation**, τόμ. 28, αρθμ. 11, σσ. 2474–2504, 14 Σεπτ. 2016, ISSN: 0899-7667. DOI: 10.1162/NECO_a_00893.

[26] P. Haueis, «The Life of the Cortical Column: Opening the Domain of Functional Architecture of the Cortex (1955–1981),» **History and Philosophy of the Life Sciences**, τόμ. 38, αρθμ. 3, 2016, ISSN: 0391-9714. DOI: 10.1007/s40656-016-0103-4. pmid: 27325058.

[27] J. Hawkins και S. Ahmad, «Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex,» **Frontiers in Neural Circuits**, τόμ. 10, 2016, ISSN: 1662-5110. DOI: 10.3389/fncir.2016.00023.

[28] ——, «Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex,» **Frontiers in Neural Circuits**, τόμ. 10, σ. 23, 2016, ISSN: 1662-5110. DOI: 10.3389/fncir.2016.00023. διεύθν.: http://journal.frontiersin.org/article/10.3389/fncir.2016.00023.

[29] Y. LeCun, «Deep learning,» CERN talk, 2016, διεύθν.: https://indico.cern.ch/event/510372/attachments/1245509/1840815/lecun-20160324-cern.pdf.

[30] S. Purdy, «Encoding Data for HTM Systems,» 18 Φεβ. 2016. arXiv: 1602.05925 [cs, q-bio]. διεύθν.: http://arxiv.org/abs/1602.05925 (επίσκεψη 02/06/2019).

[31] J. Regier, K. Pamnany, R. Giordano, R. Thomas, D. Schlegel, J. McAuliffe και Prabhat, «Learning an Astronomical Catalog of the Visible Universe through Scalable Bayesian Inference,» 10 Νοέ. 2016. arXiv: 1611.03404 [astro-ph, stat]. διεύθν.: http://arxiv.org/abs/1611.03404 (επίσκεψη 29/05/2019).

[32] C. M. Vineyard και S. J. Verzi, «Overcoming the Static Learning Bottleneck - the need for adaptive neural learning,» στο **2016 IEEE International Conference on Rebooting Computing (ICRC)**, Οκτ. 2016, σσ. 1–3. DOI: 10.1109/ICRC.2016.7738692.

[33] J. Bezanson, A. Edelman, S. Karpinski και V. Shah, «Julia: A Fresh Approach to Numerical Computing,» **SIAM Review**, τόμ. 59, αρθμ. 1, σσ. 65–98, 1 Ιαν. 2017, ISSN: 0036-1445. DOI: 10.1137/141000671.

[34] K. Brading, E. Castellani και N. Teh, «Symmetry and Symmetry Breaking,» στο **The Stanford Encyclopedia of Philosophy**, E. N. Zalta, επιμελητής, Winter 2017, Metaphysics Research Lab, Stanford University, 2017. διεύθν.: https://plato.stanford.edu/archives/win2017/entries/symmetry-breaking/ (επίσκεψη 29/05/2019).

[35] Y. Cui, S. Ahmad και J. Hawkins, «The HTM Spatial Pooler—A Neocortical Algorithm for Online Sparse Distributed Coding,» **Frontiers in Computational Neuroscience**, τόμ. 11, 2017, ISSN: 1662-5188. DOI: 10.3389/fncom.2017.00111.

[36] J. Hawkins, S. Ahmad και Y. Cui, «A Theory of How Columns in the Neocortex Enable Learning the Structure of the World,» **Frontiers in Neural Circuits**, τόμ. 11, 2017, ISSN: 1662-5110. DOI: 10.3389/fncir.2017.00081.

[37] S. Sabour, N. Frosst και G. E. Hinton, «Dynamic Routing Between Capsules,» στο **Advances in Neural Information Processing Systems 30**, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan και R. Garnett, επιμελητές, Curran Associates, Inc., 2017, σσ. 3856–3866. διεύθν.: http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.pdf (επίσκεψη 30/05/2019).

[38] V. Losing, B. Hammer και H. Wersing, «Incremental On-Line Learning: A Review and Comparison of State of the Art Algorithms,» **Neurocomputing**, τόμ. 275, σσ. 1261–1274, 31 Ιαν. 2018, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.06.084.

[39] O. Nachum, S. (Gu, H. Lee και S. Levine, «Data-Efficient Hierarchical Reinforcement Learning,» στο **Advances in Neural Information Processing Systems 31**, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi και R. Garnett, επιμελητές, Curran Associates, Inc., 2018, σσ. 3303–3313. διεύθν.: http://papers.nips.cc/paper/7591-data-efficient-hierarchical-reinforcement-learning.pdf (επίσκεψη 30/05/2019).

[40] S. Sabour, N. Frosst και G. Hinton, «Matrix Capsules with EM Routing,» **presentedat**6th International Conference on Learning Representations, ICLR, 2018.

[41] P. Xiong, Y. Zhu, Z. Sun, Z. Cao, M. Wang, Y. Zheng, J. Hou, T. Huang και Z. Que, «Application of Transfer Learning in Continuous Time Series for Anomaly Detection in Commercial Aircraft Flight Data,» στο **2018 IEEE International Conference on Smart Cloud (SmartCloud)**, Σεπτ. 2018, σσ. 13–18. DOI: 10.1109/SmartCloud.2018.00011.

[42] O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu και D. Silver. (2019). AlphaStar: Mastering the Real-Time Strategy Game StarCraft II, διεύθν.: https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/.

[43] Fabuio. (2017-07-06). Pyramidal neuron, διεύθν.: https://upload.wikimedia.org/wikipedia/commons/c/c1/Piramidal_cell.svg (επίσκεψη 31/05/2019).

[44] B. Chazelle. (). Natural Algorithms and Influence Systems, διεύθν.: https://cacm.acm.org/magazines/2012/12/157889-natural-algorithms-and-influence-systems/abstract (επίσκεψη 28/05/2019).

[45] M. Innes, **Lazy.Jl**. διεύθν.: https://github.com/MikeInnes/Lazy.jl (επίσκεψη 04/06/2019).

[46] P. Wang, «Cognitive Logic versus Mathematical Logic,»

Παράρτημα Α′

Τμήματα κώδικα

Υπολογισμός πιθανότητας σφάλματος SDR

```
N= 200_000
w= 2000
wh= 30
th= 20

lazybinom(n,k)= begin
  (k==0 || k==n) && return (1,1)
  k > n-k ?
    (k+1:n, 1:(n-k)) :
    (n-k+1:n, 1:k)
end
reducebinom(num,den,prec=20)=
  prod(BigFloat.(num, precision=prec)) / prod(BigFloat.(den,
precision=prec))
binom(n,k)= reducebinom(lazybinom(n,k)...)

ovp_set= sum( binom.(wh,th:wh) .* binom.(N-wh, wh.-(th:wh)) )
space_size= binom(N,w)
p= ovp_set / space_size
```