ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Μεταπτυχιακό: «Επιστήμη των Υπολογιστών και της Πληροφορικής»

Μελέτη προηγμένων συστημάτων υπολογιστικής νοημοσύνης και υλοποίησή τους σε Tensorflow

Κωσταντίνος Νικολάου Καραπιπεράκης

Εργασία

Που υποβλήθηκε στο τμήμα

Μηχανικών Πληροφορικής και Υπολογιστών,

της σχολής Μηχανικών,

του Πανεπιστημίου Δυτικής Αττικής,

ως μέρος των απαιτήσεων για την απόκτηση

μεταπτυχιακού διπλώματος, ειδίκευσης

Λογισμικού και Πληροφοριακών συστημάτων

Αθήνα

Σεπτέμβριος 2022

ΑΦΙΕΡΩΣΗ

Στους γονείς μου Σοφία και Νίκο και

στα παιδιά μου

Τίτλος εργασίας

Μελέτη προηγμένων συστημάτων υπολογιστικής νοημοσύνης και υλοποίησή τους σε Tensorflow

Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου και του Εισηγητή

Η πτυχιακή/διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή :

α/α	ΟΝΟΜΑ/ΕΠΩΝΥΜΟ	ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ	ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ
1	Πάρις Μαστοροκώστας	Καθηγητής	PARIS MASTOROKOSTAS Digitally signed by PARIS MASTOROKOSTAS Date: 2022.09.29 07:21:32 +03'00'
2	Αντώνιος Μπόγρης	Καθηγητής	Antonios Bogris Digitally signed by Antonios Bogris Date: 2022.09.29 11:07:05 +03'00'
3	Αναστάσιος Κεσίδης	Αναπληρωτής Καθηγητής	Anastasios Kesidis Digitally signed by Anastasios Kesidis Date: 2022.09.29 11:19:59 +03'00'

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ θερμά, τον επιβλέποντα καθηγητή του ΠΑ.Δ.Α κ. Πάρι Μαστοροκώστα, για την ευγένεια και τη βοήθεια που μου πρόσφερε. Επίσης τους καθηγητές του μεταπτυχιακού, για τις γνώσεις που μας μετέδωσαν.

ABSTRACT

Study of advanced computational intelligence systems and their implementation in Tensorflow

Computational Intelligence can also be defined as the synergy of computer systems, which lead a machine to solve problems, which are usually solved by humans, using natural intelligence. These calculations give intelligence characteristics to machines because they are based on the basic models of artificial neuron representation, which are structurally in correspondence with the neurons of our brain. In the initial chapter of the paper, reference is made to fuzzy logic systems, which are based on the theory of fuzzy sets and use participation functions for the necessary calculations. Artificial neural networks are also studied, the basic morphology of the biological neuron and its representation in the Perceptron algorithm are presented, as well as the basic architecture of neural networks. The ANFIS neurofuzzy neural network is presented and two basic neural network architectures, recurrent (RNN) and convolutional (CNN) neural networks are analyzed. In the second chapter, we analyze the core library for developing neural network architectures TensorFlow, we present Google Colaboratory, the most convenient way to use TensorFlow as well as all the core functions of the library with the code that enables them, in Python programming language. In the third chapter, two image classification models with different architectures are implemented (CNN & RNN-LSTM) and a natural language processing (NLP) model. The fourth chapter discusses the Anaconda software platform as presented by the anaconda.com website.

ΠΕΡΙΛΗΨΗ

Μελέτη προηγμένων συστημάτων υπολογιστικής νοημοσύνης και υλοποίησή τους σε Tensorflow

Η Υπολογιστική Νοημοσύνη μπορεί να οριστεί επίσης ως συνέργεια υπολογιστικών συστημάτων, τα οποία οδηγούν μία μηχανή στην επίλυση προβλημάτων, που συνήθως λύνονται από ανθρώπους, χρησιμοποιώντας φυσική νοημοσύνη. Οι υπολογισμοί αυτοί, προσδίδουν χαρακτηριστικά νοημοσύνης στις μηχανές διότι στηρίζονται στα βασικά μοντέλα αναπαράστασης τεχνητού νευρώνα, τα οποία δομικά, είναι σε αντιστοιχία με τους νευρώνες του εγκεφάλου μας. Στο αρχικό κεφάλαιο της εργασίας, γίνεται αναφορά στα συστήματα ασαφούς λογικής, τα οποία βασίζονται στη θεωρία των ασαφών συνόλων και χρησιμοποιούν συναρτήσεις συμμετοχής για τους απαραίτητους υπολογισμούς. Επίσης γίνεται μελέτη στα τεχνητά νευρωνικά δίκτυα, παρουσιάζεται η βασική μορφολογία του βιολογικού νευρώνα και η απεικόνισή του στον αλγόριθμο του Perceptron, αλλά και στη βασική αρχιτεκτονική των νευρωνικών δικτύων. Παρουσιάζεται το νευροασαφές νευρωνικό δίκτυο ANFIS και αναλύονται δύο βασικές αρχιτεκτονικές των νευρωνικών δικτύων, τα αναδρομικά(RNN) και τα συνελικτικά(CNN) νευρωνικά δίκτυα. Στο δεύτερο κεφάλαιο, αναλύουμε τη βασική βιβλιοθήκη για την ανάπτυξη αρχιτεκτονικών των νευρωνικών δικτύων το TensorFlow, παρουσιάζουμε το Google Colaboratory, τον πιο βολικό τρόπο χρήσης του TensorFlow καθώς και όλες τις βασικές λειτουργίες της βιβλιοθήκης με τον κώδικα που τις ενεργοποιεί, σε γλώσσα προγραμματισμού Python. Στο τρίτο κεφάλαιο υλοποιούνται δύο μοντέλα ταξινόμησης εικόνων με διαφορετικές αρχιτεκτονικές (CNN & RNN-LSTM) και ένα μοντέλο επεξεργασίας φυσικής γλώσσας (NLP). Στο τέταρτο κεφάλαιο αναλύεται η πλατφόρμα λογισμικού Anaconda όπως παρουσιάζεται από τον ιστότοπο anaconda.com.

ΚΑΤΑΛΟΓΟΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΕΙΣΑΓΩΓΗ

Η ιδέα της τεχνητής νοημοσύνης, που συνεπάγεται μηχανή που μπορεί να σκέφτεται χωρίς ανθρώπινη βοήθεια, είναι εκπληκτικά παλιά. Μπορεί να χρονολογηθεί στις ινδικές φιλοσοφίες του Charvaka, περίπου από το 1.500 π.Χ. Η βάση της τεχνητής νοημοσύνης είναι η φιλοσοφική ιδέα ότι η ανθρώπινη λογική μπορεί να χαρτογραφηθεί σε μία μηχανική διαδικασία. Ο εγκέφαλος είναι το πιο σημαντικό όργανο του ανθρώπινου σώματος. Είναι η κεντρική μονάδα επεξεργασίας για όλες τις λειτουργίες που εκτελούμε. Με βάρος μόνο 1,5 κιλό, έχει περίπου 86 δισεκατομμύρια νευρώνες. Ένας νευρώνας ορίζεται ως ένα κύτταρο που μεταδίδει νευρικές ώσεις ή ηλεκτροχημικά σήματα. Ο εγκέφαλος είναι ένα πολύπλοκο δίκτυο νευρώνων που επεξεργάζεται πληροφορίες μέσω ενός συστήματος πολλών διασυνδεδεμένων νευρώνων. Τα ερεθίσματα που λαμβάνει από εξωτερικούς παράγοντες, τροφοδοτούν μία διαδικασία αναγνώρισης σήματος, αξιολόγησης και εξαγωγής αποτελέσματος. Ήταν πάντα δύσκολο να κατανοήσουμε τις λειτουργίες του εγκεφάλου. Ένα αποφασιστικό βήμα, προήλθε κυρίως από τη χρήση των λεγόμενων τεχνητών νευρωνικών δικτύων **(** ANN **),** που ξεκινώντας από τους μηχανισμούς που ρυθμίζουν τα φυσικά νευρωνικά δίκτυα, σχεδιάζουν να προσομοιώσουν την ανθρώπινη σκέψη. Λόγω της προόδου στις τεχνολογίες υπολογιστών, μπορούμε πλέον να προγραμματίζουμε τα τεχνητά νευρωνικά δίκτυα. Η πειθαρχία των ANNs, προέκυψε από τη σκέψη να μιμηθεί τη λειτουργία του ίδιου ανθρώπινου εγκεφάλου που προσπαθούσε να λύσει το πρόβλημα. Τα μειονεκτήματα των συμβατικών προσεγγίσεων και οι διαδοχικές εφαρμογές τους έχουν ξεπεραστεί σε καλά καθορισμένα τεχνικά περιβάλλοντα. Υπάρχει πολύς δρόμος ακόμα να καλυφθεί, αν σκεφτούμε ότι ένα μικρό παιδί μπορεί να αναγνωρίσει τη μαμά του μέσα σε ένα τεράστιο πλήθος, αλλά ένας υπολογιστής με κεντρική αρχιτεκτονική δεν θα μπορούσε να κάνει το ίδιο τόσο αποτελεσματικά. Η λειτουργία ολόκληρου του νευρωνικού δικτύου είναι απλώς ο υπολογισμός των εξόδων όλων των νευρώνων. Ουσιαστικά, το ANN είναι ένα σύνολο μαθηματικών προσεγγίσεων μέσω συναρτήσεων. Το λογισμικό μπορεί τώρα να μιμηθεί τους μηχανισμούς που απαιτούνται για να κερδίσει για παράδειγμα έναν αγώνα σκακιού ή για να μεταφράσει κείμενο σε διαφορετική γλώσσα σύμφωνα με τους γραμματικούς του κανόνες.

ΚΕΦΑΛΑΙΟ 1ο

ΥΠΟΛΟΓΙΣΤΙΚΗ ΝΟΗΜΟΣΥΝΗ (Computational Intelligence - CI)

Ο τομέας της υπολογιστικής νοημοσύνης συγκεντρώνει ένα σύνολο υπολογιστικών μεθοδολογιών και προσεγγίσεων, που εμπνέονται από τη διανοητική λειτουργία του ανθρώπου και τη φύση. Επιχειρεί την αντιμετώπιση πολύπλοκων προβλημάτων του πραγματικού κόσμου, στα οποία η μαθηματική ή, η παραδοσιακή σαφής (Bolean) λογική δεν μπορεί να δώσει λύσεις με σχετική ακρίβεια. Στο πεδίο της υπολογιστικής νοημοσύνης (CI) εμπεριέχεται ο κλάδος της πληροφορικής που ασχολείται με το σχεδιασμό και την υλοποίηση εκείνων των υπολογιστικών συστημάτων, που μιμούνται στοιχεία της ανθρώπινης σκέψης, συμπεριφοράς και βιολογίας. Υπολογιστικά συστήματα τα οποία αναπτύσσονται με συγκεκριμένες διαδικασίες ώστε να επιλύουν προβλήματα, να εξάγουν συμπεράσματα, να προσαρμόζονται στις συνθήκες ή να κάνουν προβλέψεις και να αναγνωρίζουν πρότυπα. Ο τομέας της υπολογιστικής νοημοσύνης μπορεί να διαχωριστεί:

- Στην Εξελικτική Υπολογιστική (Evolutionary Computation - EC), η οποία διακρίνεται για τη μελέτη των έμβιων οργανισμών. Εξετάζει έννοιες όπως της μετάλλαξης , του πληθυσμού και της φυσικής επιλογής, για την επίλυση των προβλημάτων (GA).
- Στα Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks - ANNs) τα οποία προσομοιάζουν τη λειτουργία των νευρώνων του ανθρώπινου εγκεφάλου. Αναπτύσσονται διαφορετικά νευρωνικά δίκτυα, ανάλογα το πεδίο της εγκεφαλικής διεργασίας που πρέπει να ενεργοποιηθεί (εικόνα, λόγος κλπ)
- Στα Συστήματα Ασαφούς Λογικής (Fuzzy Logic Systems - FLS), τα οποία βασίζονται σε καταστάσεις ή αντικείμενα (ασαφή σύνολα) μη-ευκρινώς (non crisp) διαχωρισμένα.

Τα συστήματα Τεχνητής Νοημοσύνης ακολουθούν τις αρχές της δυαδικής λογικής και αντίστοιχα τα συστήματα Υπολογιστικής Νοημοσύνης τις αρχές της ασαφούς λογικής με την οποία μπορεί να αντιμετωπιστεί η απουσία, αλλά και η άγνοια των δεδομένων σε ένα μοντέλο διεργασίας, σε αντίθεση με την Τεχνητή Νοημοσύνη, η οποία απαιτεί ακριβείς γνώσεις. Στο παρόν, γίνεται εξέταση στα συστήματα ασαφούς λογικής και νευρωνικών δικτύων.

1.1 Συστήματα Ασαφούς λογικής

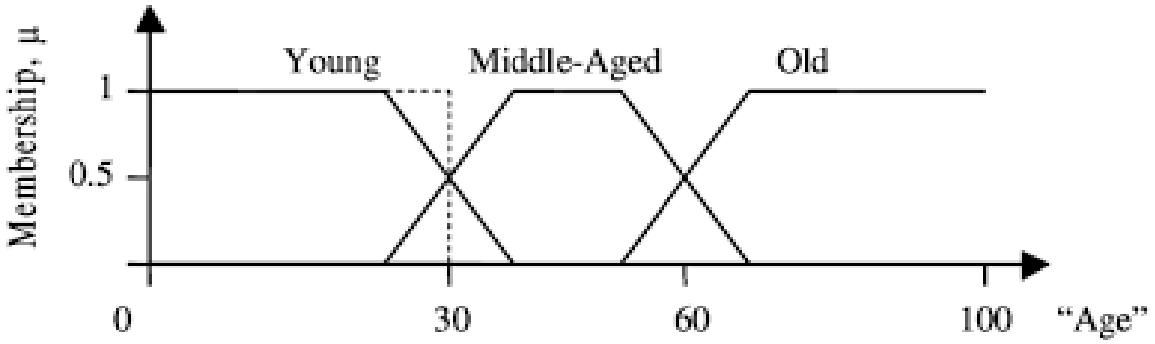
Τα συστήματα ασαφούς λογικής βασίζονται στη θεωρία των ασαφών συνόλων, η οποία προτάθηκε το 1965 από τον καθηγητή Lotfi A. Zadeh στο Πανεπιστήμιο Berkeley της Καλιφόρνια και αποτελεί γενίκευση των κλασικών συνόλων. Μέχρι τότε, η σχέση μεταξύ μεμονωμένων στοιχείων κι ενός συνόλου, αποτυπωνόταν με τις τιμές {0,1}, στο πεδίο δηλαδή της Boolean λογικής, με βάση την οποία ένα στοιχείο ανήκει (1) ή δεν ανήκει (0), στο σύνολο που εξετάζουμε. Η ασαφής θεωρία του Zadeh, εξετάζει τον βαθμό συμμετοχής ενός στοιχείου σε ένα ή περισσότερα ασαφή σύνολα. Τα σύνολα αυτά βασίζονται σε λεκτικές περιγραφές που προσιδιάζουν στον τρόπο αντίληψης του ανθρώπου. Η θεωρία των ασαφών συνόλων παρέχει έναν μηχανισμό για την αναπαράσταση γλωσσικών κατασκευών όπως «πολλά», «χαμηλά», «μεσαία», «συχνά», «λίγα». Μία πρόταση μπορεί να μην είναι (απολύτως) αληθής, ούτε (απολύτως) ψευδής, αλλά να χαρακτηρίζεται από κάποιο βαθμό αλήθειας ή ψεύδους. Η "αλήθεια" μίας πρότασης αν το εξετάσουμε λεπτομερώς, συνήθως δεν είναι δίτιμη {0,1}, αλλά είναι πλειότιμη στο διάστημα [0,1]. Τα ασαφή συστήματα επιτρέπουν στα στοιχεία να βρίσκονται εν μέρει σε ένα σύνολο. Σε κάθε στοιχείο μπορεί να δοθεί ένας βαθμός συμμετοχής (από 0 έως 1), ο οποίος ορίζεται υποκειμενικά από κάποιο ειδικό ή μετά από μετρήσεις. Η τιμή του εκφράζεται μέσω της χαρακτηριστικής συνάρτησης ή αλλιώς συνάρτηση συμμετοχής και δηλώνει το βαθμό συμμετοχής ενός στοιχείου x στο ασαφές σύνολο A. Αν Ω καλούμε ένα κλασικό σύνολο (crisp set), μία συλλογή αντικειμένων την οποία μπορούμε να ονομάσουμε υπερ-σύνολο αναφοράς (universe of discourse), τότε ως ασαφές υποσύνολο A του Ω, ορίζεται το σύνολο των διατεταγμένων ζευγών:

$$A = \{(x_i, \mu A(x_i)): \mu A(x_i) \in [0,1], \forall x_i \in \Omega]\}, \quad i=1,2,3....N$$

όπου μA(x) μία συνάρτηση από το Ω στο διάστημα [0,1], η οποία ονομάζεται χαρακτηριστική συνάρτηση A: $\Omega \rightarrow [0,1]$ ή αλλιώς συνάρτηση συμμετοχής (Membership Function-MF) του ασαφούς υποσυνόλου A. Η συνάρτηση συμμετοχής μA(x) συμβολίζεται πιο απλά ως A(x) και η τιμή της καλείται βαθμός συμμετοχής (membership grade) του x στο σύνολο A. Η βάση της ασαφούς λογικής εδράζεται στην ανάγκη του ανθρώπου για επικοινωνία. Οι σκέψεις ενός ατόμου είναι ξεκάθαρες (crisp) για τον ίδιο. Όταν όμως χρειαστεί να μεταδώσει, να μεταφέρει τις σκέψεις του, να επικοινωνήσει με άλλα άτομα, τα οποία έχουν διαφορετικό λειτουργικό σύστημα , παύουν να θεωρούνται τόσο ξεκάθαρες.

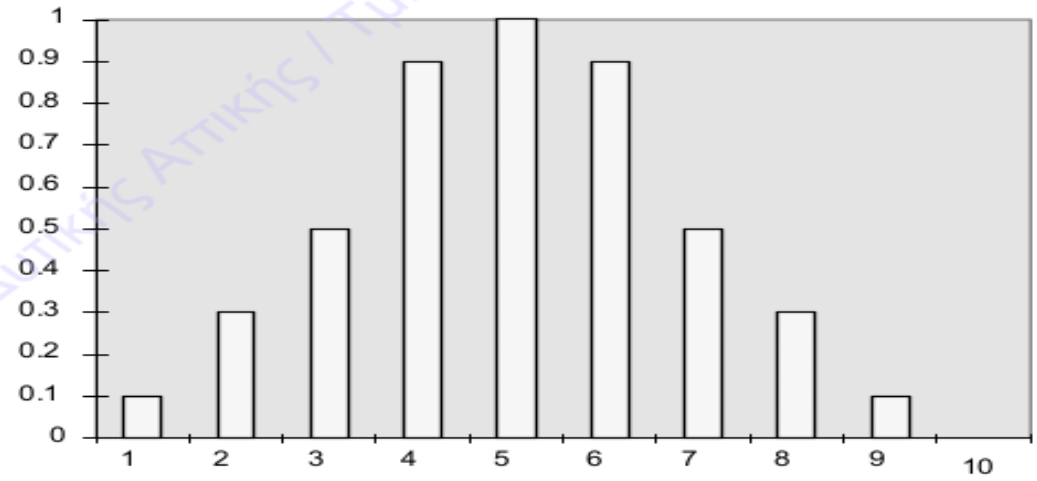
Έτσι ένας άνθρωπος 30 ετών, μπορεί να ταξινομηθεί από κάποιον που είναι για παράδειγμα 45 ετών, ότι ανήκει στο ασαφές σύνολο "νέος" ή να ταξινομηθεί από έναν 15χρονο, στο

ασαφές σύνολο “μεσήλικας”. Βλέπουμε στο κάτωθι διάγραμμα ότι ο βαθμός συμμετοχής της λεκτικής μεταβλητής Age = “30 ετών”, στα ασαφή σύνολα Young και Middle-Aged είναι 0.5.



Σχήμα 1 : Συναρτήσεις συμμετοχής ασαφών συνόλων της λεκτικής μεταβλητής “Age”.

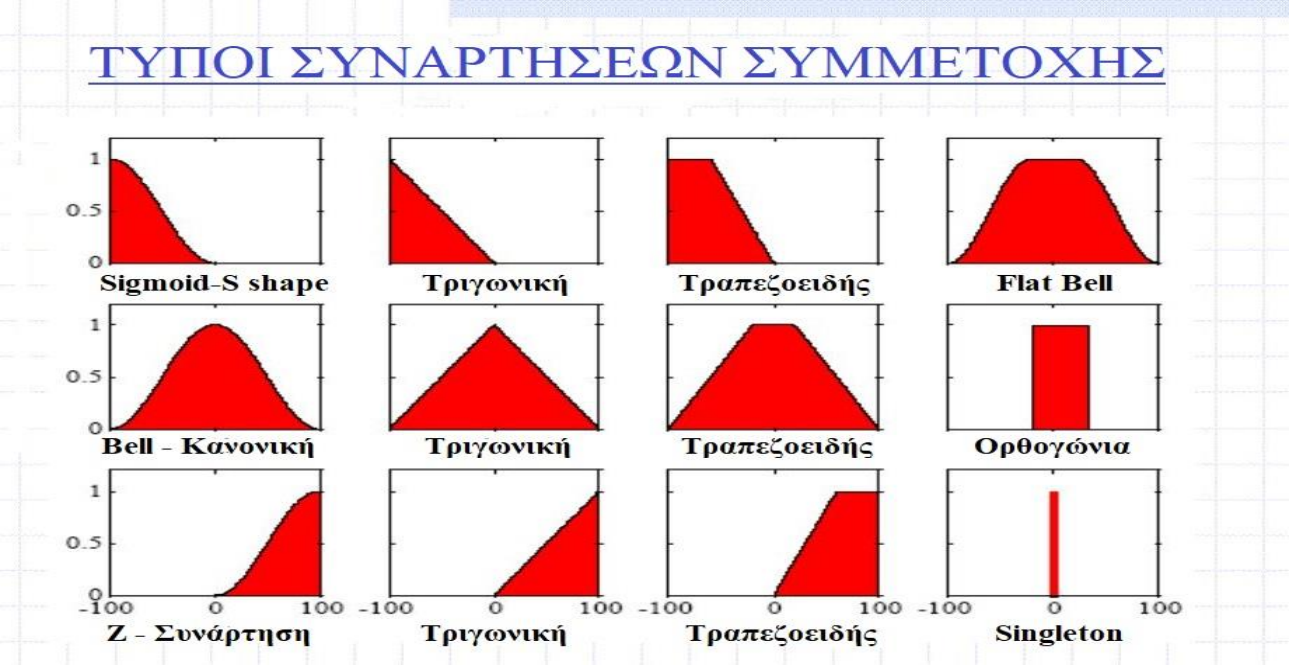
Στο ανωτέρω διάγραμμα το πεδίο τιμών (universe of discourse) είναι συνεχές, δηλαδή η “Age” ∈ Ω =[0 , 100] αποτελεί διάστημα του πραγματικού χώρου και όποιο ασαφές σύνολο προκύπτει ονομάζεται συνεχές : A = {∫ A(x) /x : A(x) ∈ [0,1], ∀ x ∈ Ω : Ω=[xmin , xmax]. Τα ασαφή σύνολα των οποίων το πεδίο ορισμού τους είναι διακριτό, περιγράφονται με ασυνεχείς συναρτήσεις και ονομάζονται διακριτά. Για το ασαφές σύνολο Α, όταν το Ω είναι διακριτό, δηλαδή Ω = [x1,x2,...,xn] ισχύει το εξής :A = μA(x1)/x1 + μA(x2)/x2 + ⋯ + μA(xn)/xn = ΣμA(xi)/xi, (ένωση ζευγών singleton). Ο τελεστής συν (+) και το ολοκλήρωμα, αναφέρονται στην ένωση συνόλων ή διανυσμάτων.



Σχήμα 2 : Συναρτήσεις συμμετοχής διακριτού ασαφούς συνόλου.

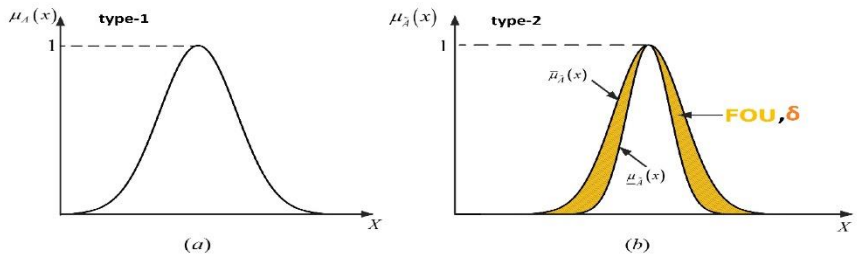
1.1.1 Οι συναρτήσεις συμμετοχής (Membership Functions-MF)

Περιγράφουν πλήρως ένα ασαφές σύνολο. Όταν το πεδίο ορισμού (universe of discourse) είναι συνεχές, θεωρείται πρακτικά ανέφικτο να περιγράψουμε τη συνάρτηση με βάση τα ζεύγη Singleton. Οι συνεχείς MF χρησιμοποιούν παραμέτρους και ανάλογα τον τύπο που έχουν, αποκαλύπτουν τη μορφή της συνάρτησης τους. Η επιλογή της MF γίνεται ανάλογα με το πρόβλημα. Ενδεικτικά παρουσιάζονται συναρτήσεις συμμετοχής ευρείας χρήσης.



Σχήμα 3 : Διάφοροι τύποι και γραφήματα, συναρτήσεων συμμετοχής.

Μία χαρακτηριστική συνάρτηση A: Ω → [0,1] ενός ασαφούς συνόλου A ως προς το σύνολο αναφοράς Ω, απεικονίζει ένα συγκεκριμένο x ∈ Ω, σε μία τιμή στο διάστημα [0,1] και ονομάζεται ασαφές σύνολο τύπου-1 (type-1). Αρκετοί ερευνητές θεώρησαν ότι τα ασαφή σύνολα τύπου-1 συχνά περιορίζουν την ασάφεια. Έτσι προτάθηκε η χαρακτηριστική συνάρτηση να λαμβάνει όχι μία τιμή, αλλά τιμές από ένα σύνολο κλειστών διαστημάτων, τα οποία είναι υποσύνολα του διαστήματος [0,1]. Τέτοια ασαφή σύνολα ονομάζονται τύπου-2 (type-2).



1.1.2 Τεχνικά χαρακτηριστικά των ασαφών συνόλων

- Σύνολο υποστήριξης (support) ή ενεργός περιοχή ή στήριγμα ενός ασαφούς συνόλου Α ονομάζουμε το κλασσικό (crisp-σαφές) σύνολο, το οποίο περιέχει όλα τα στοιχεία x του Ω για τα οποία η τιμή της συνάρτησης συμμετοχής τους είναι μεγαλύτερη από το μηδέν. Δηλαδή ισχύει : Supp (A) = { x ∈ Ω : A(x) > 0 }.
- Σύνολο α-διατομής (α-cut set) για ένα ασαφές σύνολο Α με σύνολο αναφοράς Ω ονομάζουμε α- διατομή το σαφές σύνολο Αα : Αα = { x ∈ Ω : A(x) ≥ α }, α∈[0,1].
- Πυρήνας ασαφούς συνόλου ορίζονται τα στοιχεία x ∈ Ω για τα οποία ισχύει A(x) = 1. Core(A)= { x ∈ Ω : A(x) =1}. Επίσης ονομάζεται πυρήνας το σύνολο α-διατομής για α=1.
- Ύψος (height) ασαφούς συνόλου είναι η μέγιστη τιμή της A(x), συνάρτησης συμμετοχής, Height(A) = maxA(x).
- Κανονικό (normal fuzzy set) ασαφές σύνολο , ονομάζουμε αυτό που έχει μία τουλάχιστον συνάρτηση συμμετοχής με τιμή 1. A(x)=1.
- Ασαφές Singleton καλούμε το ασαφές σύνολο που το σύνολο υποστήριξής του, έχει ένα μόνο στοιχείο, για το οποίο ισχύει : A(x_0)=1 και A(x) = 0, ∀ x_0 ≠ x, x ∈ Ω.
- Σημεία καμπής (crossover points) καλούνται τα σημεία για τα οποίο ισχύει : A(x) =0.5.
- Μέγεθος (cardinality) ασαφούς συνόλου ορίζεται το άθροισμα των βαθμών συμμετοχής των σημείων του συνόλου αναφοράς. |A|= Σ A(x).
- Τελεστές min - max οι οποίοι χρησιμοποιούνται για την εύρεση του ελάχιστου (min) ή του μέγιστου (max) μεταξύ δύο στοιχείων. Περιγράφονται ως, $\mu_1 \wedge \mu_2$ = min (μ_1 , μ_2) και $\mu_1 \vee \mu_2$ = max (μ_1 , μ_2). Οι τελεστές min - max είναι ανάλογοι των αλγεβρικών τελεστών (*) και (+) αντίστοιχα. Επίσης διαθέτουν την επιμεριστική και προσεταιριστική ιδιότητα. Γίνεται ευρεία χρήση αυτών των τελεστών κατά τη σύνθεση των ασαφών σχέσεων που μοιάζει με πολλαπλασιασμό πινάκων, με τη διαφορά ότι η πράξη της πρόσθεσης αντικαθίσταται με το max, δηλαδή το ασαφές

OR και η πράξη του πολλαπλασιασμού των στοιχείων αντικαθίσταται με το min, δηλαδή το ασαφές AND.

- Οι λεκτικές μεταβλητές αποτελούν γενικεύσεις των κλασσικών μεταβλητών και δέχονται ως τιμές λέξεις, οι οποίες περιγράφονται από ασαφή σύνολα, ορισμένα στο σύνολο αναφοράς (universe of discourse). Η λεκτική μεταβλητή "ηλικία" για παράδειγμα, ορίζεται στο διάστημα $\Omega = [0 , 100]$ και παίρνει τρείς ασαφής τιμές: "νέος" , "μεσήλικας", "ηλικιωμένος". Κάθε τιμή περιγράφεται από το αντίστοιχο ασαφές σύνολο $A_{νέος}(x)$, $A_{μεσηλ}(x)$, $A_{ηλικ}(x)$, $x \in \Omega$.
- Ασαφείς κανόνες θεωρούνται οι υποθετικές προτάσεις της μορφής : *IF x is A THEN y is B.* Οι εκφράσεις *x is A* και *y is B* θεωρούνται ασαφής προτάσεις, ενώ όροι *A και B καλούνται* λεκτικές τιμές (ασαφή σύνολα) των x και y, με πεδίο ορισμού *X* και *Y* αντίστοιχα. Το αριστερό τμήμα του κανόνα *(IF* <fuzzy proposition>*),* το ονομάζουμε τμήμα υπόθεσης (pre-conditional part, IF-part) και το δεξιό τμήμα *(THEN* <fuzzy proposition>*),* το ονομάζουμε τμήμα απόδοσης ή συμπεράσματος (conseqent, THEN-part). Για συντομία ο κανόνας *IF x is A THEN y is B* γράφεται ως $A \rightarrow B$. Η ασαφής σχέση που ο κανόνας ορίζει μεταξύ x και y, αναφέρεται ως σχέση συμπερασμού. Αν A και B ασαφή σύνολα, στους αντίστοιχους χώρους X, Y και οι ασαφής προτάσεις p: *x is A* και q: *y is B* , τότε η εκπλήρωση του κανόνα *IF p THEN q,* εξαρτάται από τον βαθμό αλήθειας ή το βαθμό εκπλήρωσης της *p* και συμπίπτει με το βαθμό συμμετοχής του x στο ασαφές σύνολο A. Η συνάρτηση αλήθειας (Truth function) της *p* μπορεί να αποτυπωθεί συμβολικά ως εξής : $T(p) = T(x \text{ is } A) = A(x)$, $A(x) \in [0,1]$, $T : x \in X \rightarrow [0,1]$. Η ασαφής σχέση $p \rightarrow q$ περιγράφεται : $R = p \rightarrow q = A \rightarrow B = \int R(x,y)/(x,y)$, $R(x,y) = \gamma [A(x) , B(y)]$, η συνάρτηση $\gamma[.,.,]$ καλείται συνάρτηση συμπερασμού και καθορίζει τη συνάρτηση συμμετοχής της σχέσης $A \rightarrow B$ με βάση $A(x) , B(y)$.

1.1.3 Μορφές Ασαφών Συστημάτων

Τα ασαφή συστήματα διακρίνονται ανάλογα με τη μορφή των κανόνων. Συχνότερα συναντάμε τα κάτωθι :

- Τύπου Mamdami, προσδιορίζεται ως ένας κανόνας της μορφής: If x is A then y is B. Οι έξοδοι των κανόνων αυτής της μορφής είναι ασαφή σύνολα.

- Τύπου Sugeno-Tagaki, αποτελεί ένα κανόνα της μορφής : If x is A then y is c. Το c είναι αριθμός ή ένα ασαφές σύνολο.
- Τύπου Sugeno-Tagaki-Kang, αποτελεί έναν από τους κυριότερους τύπους ασαφούς κανόνα και προσδιορίζεται από τη μορφή: If x is A then y is $c_0 + c_1 x$, όπου $c_0, c_1 \in R$. Οι έξοδοι των κανόνων αυτής της μορφής είναι συναρτήσεις των εισόδων.

Η συμπεριφορά ενός ασαφούς συστήματος χαρακτηρίζεται από ένα σύνολο γλωσσικών κανόνων που αποτελεί τη βάση κανόνων. Ένας τυπικός γλωσσικός κανόνας έχει την ακόλουθη μορφή: IF x_1 is A_1 and x_2 is A_2 THEN y is B.

1.1.4 Τα βασικά στοιχεία ενός ασαφούς συστήματος

Τα στοιχεία που χρησιμοποιούνται στη λύση των όποιων προβλημάτων, είναι τα ακόλουθα:

◊ Ασαφοποίηση (fuzzification), μετατροπή δεδομένων εισόδου σε ασαφή στοιχεία.
◊ Ανάπτυξη των κανόνων , βάση κανόνων.
◊ Επεξεργασία των κανόνων, ασαφής συμπερασμός.
◊ Απασαφοποίηση (defuzzification), μετατροπή των αποτελεσμάτων από ασαφή σε ευκρινή , αριθμητικά στοιχεία.

A) Ασαφοποίηση (Fuzzification)

Ασαφοποίηση θεωρείται η διαδικασία μετατροπής μίας ευκρινούς τιμής εισόδου σε μία ασαφή τιμή που πραγματοποιείται με τη χρήση των πληροφοριών στη βάση γνώσεων. Οι καθαρές είσοδοι (crisp inputs) που αντιστοιχούν στις μεταβλητές, αντικαθίστανται από λεκτικούς όρους ή προτάσεις και δημιουργούνται συναρτήσεις συμμετοχής αντίστοιχα για κάθε κριτήριο. Οι MFs παίρνουν την τιμή 1 σε ένα σημείο ή σε διάστημα, αριστερά του οποίου είναι αύξουσες και δεξιά φθίνουσες. Αν και στη βιβλιογραφία μπορούν να παρατηρηθούν διάφοροι τύποι καμπυλών, οι Gaussian, τριγωνικές, σιγμοειδείς και τραπεζοειδείς MFs, είναι οι πιο συχνά χρησιμοποιούμενες στη διαδικασία ασαφοποίησης. Οι τύποι των συγκεκριμένων MFs, μπορούν εύκολα να εφαρμοστούν από ενσωματωμένους ελεγκτές. Οι συναρτήσεις συμμετοχής ορίζονται μαθηματικά με διάφορες παραμέτρους, οι οποίες μπορούν να προσαρμοστούν κατάλληλα, για τη βελτίωση της απόδοσης ενός ασαφούς ελεγκτή.

Β) Βάση κανόνων

Σε αυτό το βήμα, οι γνώσεις των εμπειρογνωμόνων διατυπώνονται σε ένα πεπερασμένο αριθμό κανόνων. Η βάση κανόνων περιέχει τους κανόνες που πρέπει να χρησιμοποιούνται για τη λήψη αποφάσεων. Αυτοί οι κανόνες βασίζονται γενικά στην προσωπική εμπειρία του "ειδικού" και τη διαίσθηση. Ωστόσο, σε ορισμένες περιπτώσεις, οι κανόνες μπορούν να ληφθούν χρησιμοποιώντας νευρωνικά δίκτυα, γενετικούς αλγορίθμους ή μερικές εμπειρικές προσεγγίσεις. Ένας κανόνας αποτελείται από δύο κύρια μέρη: ένα προηγούμενο μπλοκ (μεταξύ του If και Then) και ένα συνακόλουθο μπλοκ (μετά το Then). Αν και το προηγούμενο και τα συνακόλουθα μέρη έχουν μεμονωμένα ορίσματα στα παραπάνω, ένας κανόνας μπορεί να γραφτεί με πολλαπλά ορίσματα. Αυτή η διεργασία αξιολόγησης των κανόνων συντελείται σε τρία στάδια: τη συσσώρευση (aggregation), τη σημαντικότητα (implication) και τη συγκέντρωση (accumulation). Στη φάση της συσσώρευσης αξιολογείται η εκπλήρωση κάθε κανόνα, από το βαθμό εκπλήρωσης των μεταβλητών του και επιχειρείται η επιλογή κατάλληλων τελεστών για το πρόβλημα. Στη φάση της σημαντικότητας εκτιμάται η βεβαιότητα του συνόλου των κανόνων, εξετάζοντας κάθε κανόνα χωριστά. Στην περίπτωση όπου υπάρχει σίγουρο αποτέλεσμα, ο "ειδικός" δίνει ως τιμή τη μονάδα (1). Γίνεται έτσι η σύνδεση στα ποσοστά βεβαιότητας του κάθε κανόνα με τον τελικό βαθμό εκπλήρωσής του. Στην τελευταία φάση της συγκέντρωσης, αξιολογούνται οι κανόνες που προσδίδουν παρόμοια αποτελέσματα με διαφορετικό βαθμό εκπλήρωσης. Ομαδοποιούνται οι κανόνες και αναπροσαρμόζονται, ώστε να υπάρχει ένας βαθμός εκπλήρωσης ανά ομάδα κανόνων, με τη βοήθεια κατάλληλων τελεστών.

Γ) Επεξεργασία και εξαγωγή συμπεράσματος

Σε αυτήν τη διαδικασία τα ασαφή συμπεράσματα εξάγονται χρησιμοποιώντας τους κανόνες στη βάση κανόνων. Κατά τη διάρκεια αυτής της διαδικασίας, κάθε κανόνας αξιολογείται ξεχωριστά και στη συνέχεια λαμβάνεται μία απόφαση για κάθε μεμονωμένο κανόνα. Το αποτέλεσμα είναι ένα σύνολο ασαφών αποφάσεων. Λογικοί τελεστές, όπως "AND", "OR" και "NOT" καθορίζουν τον τρόπο συνδυασμού των ασαφών μεταβλητών.

Δ) Από-ασαφοποίηση (Defuzzification)

Το τελευταίο βήμα αφορά μία διαδικασία όπου η ασαφής έξοδος μεταφράζεται σε μία ενιαία ευκρινή τιμή, με γνώμονα τον βαθμό των τιμών των συναρτήσεων συμμετοχής. Η διαδικασία defuzzification είναι ένας αντίστροφος μετασχηματισμός σε σύγκριση με τη διαδικασία

fuzzification, διότι σε αυτή τη διαδικασία, η ασαφής έξοδος μετατρέπεται στις ευκρινείς τιμές που θα εφαρμοστούν στο σύστημα. Υπάρχουν αρκετές ευρετικές μέθοδοι από-ασαφοποίησης. Οι πλέον διαδεδομένες μέθοδοι αυτής της μετατροπής είναι της μέγιστης τιμής, η οποία περιλαμβάνει τις περιπτώσεις από-ασαφοποίησης: α) του μικρότερου των μεγίστων (Smallest of maxima – SOM), β) μεγαλύτερου των μεγίστων (Largest of maxima – LOM), γ) του μέσου των μεγίστων (Middle of maxima – MOM) και αυτή του κέντρου βάρους (Center of area - Centroid).

1.2 Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks - ANNs)

Τα τεχνητά νευρικά δίκτυα (ANNs) είναι ένα σύνολο αλγορίθμων, που η δομή τους και η λειτουργία τους, εμπνέονται σε γενικές γραμμές από τη γνώση που έχουμε για τους βιολογικούς εγκεφάλους. Τα ANNs αποτελούνται από απλές μονάδες (νευρώνες) που συνδέονται μεταξύ τους. Αντίστοιχα στη βιολογία, οι φυσικοί νευρώνες λαμβάνουν, επεξεργάζονται και μεταδίδουν ένα σήμα σε άλλους νευρώνες, ενεργώντας σαν διακόπτης. Τα στοιχεία ενός νευρωνικού δικτύου είναι αρκετά απλά από μόνα τους. Η πολυπλοκότητα και η ισχύς αυτών των συστημάτων προέρχονται από την αλληλεπίδραση μεταξύ των στοιχείων. Ένας ανθρώπινος εγκέφαλος θεωρείται ότι έχει περίπου 100 δισεκατομμύρια νευρώνες και 100 τρισεκατομμύρια συνδέσεις (συνάψεις). Ως εκ τούτου, το μαθηματικό μοντέλο των ANNs, είναι εμπνευσμένο από τους βιολογικούς νευρώνες.

1.2.1 Βιολογική μορφολογία νευρώνα

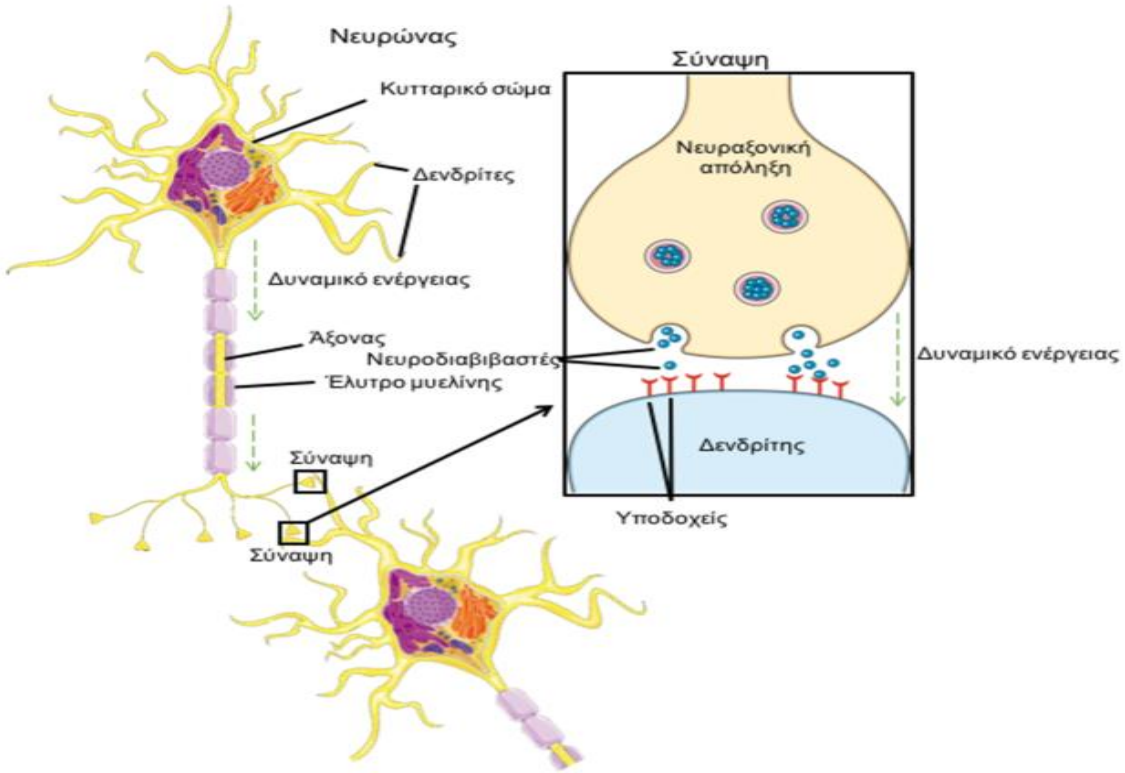
Η βιολογική νευρωνική μορφολογία (δομή) αποτελεί τη βάση για τη μελέτη υπολογιστικών νευρωνικών δικτύων. Το βασικό δομικό στοιχείο του κεντρικού νευρικού συστήματος (ΚΝΣ) είναι ο νευρώνας (Σχ.5), τον οποίο διατρέχουν βιοχημικά σήματα. Αποτελεί τον μηχανισμό που επεξεργάζεται και μεταδίδει πληροφορίες από και προς τα διάφορα μέρη του νευρικού συστήματος. Από άποψη επεξεργασίας πληροφοριών, ένας νευρώνας εκτελεί τα ακόλουθα τρία βήματα:

(i) οι δενδρίτες λαμβάνουν τα βιοχημικά σήματα (πληροφορίες), μέσω των συνάψεων συνδεδεμένων νευρώνων και τις προωθούν προς το κυτταρικό σώμα,

(ii) το κυτταρικό σώμα υποδέχεται τα σήματα ενεργώντας περαιτέρω επεξεργασία πληροφοριών και

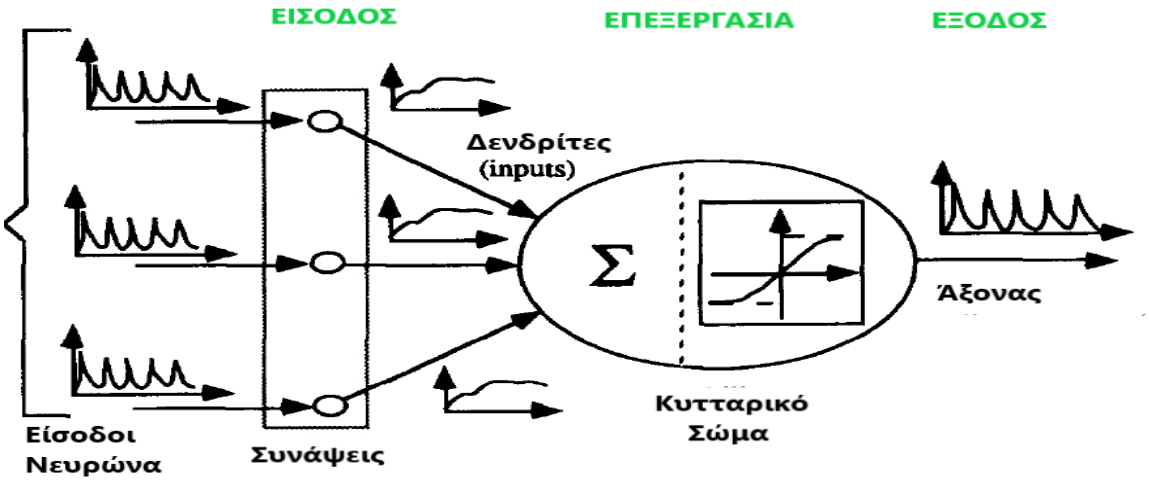
(iii) ο νευρώνας μεταδίδει αυτές τις πληροφορίες σε άλλους νευρώνες μέσω μίας μοναδικής εξόδου που ονομάζεται άξονας.

Το σημείο σύνδεσης ενός άξονα με δενδρίτη, ονομάζεται σύναψη. Οι συνάψεις είναι χώρος αποθήκευσης του παρελθόντος. Παρέχουν μακροχρόνια μνήμη και εμπειρία, λειτουργώντας περίπου σαν βάσεις γνώσεων. Ένας μοναδικός βιολογικός νευρώνας μπορεί να έχει κατά μέσο όρο, περί τις 10.000 συναπτικές συνδέσεις. Από θεωρητική άποψη, ο νευρώνας μπορεί να θεωρηθεί ως σύστημα πολλαπλών εισόδων-μονής εξόδου.



Σχήμα 5. Αναπαράσταση βιολογικού νευρώνα

Όταν η επεξεργασμένη πληροφορία (βιοχημικό σήμα) καταλήξει στη συναπτική διασταύρωση στην έξοδο του άξονα, αναλαμβάνουν οι νευροδιαβιβαστές να τη μεταβιβάσουν στον δενδρίτη του νευρώνα υποδοχής, προκαλώντας ηλεκτρική απόκριση. Ανάλογα με την υπέρβαση ή όχι ενός ορίου, η ένταση της απόκρισης μπορεί να είναι διεγερτική ή ανασταλτική.



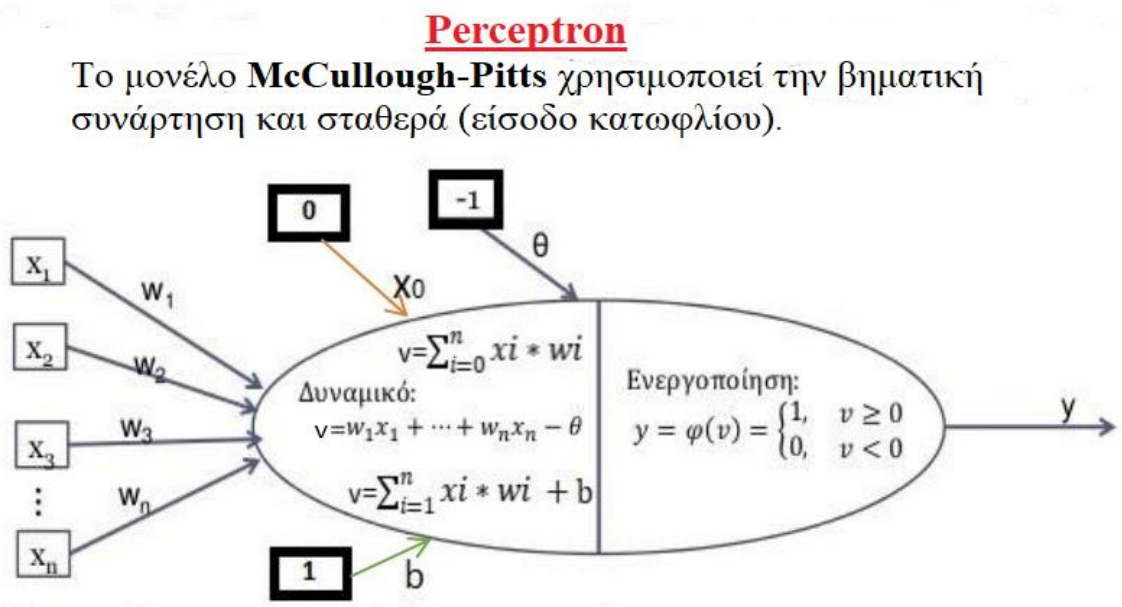
Σχήμα 6. Απλοποιημένο μοντέλο των χαρακτηριστικών επεξεργασίας σήματος ενός βιολογικού νευρώνα.

Ο βιολογικός νευρώνας έχει δύο βασικά στοιχεία, τη σύναψη και το κυτταρικό σώμα. Αυτά διενεργούν όλες τις υπολογιστικές εργασίες, όπως απόκτηση γνώσεων, μάθηση-εμπειρία και αναγνώριση προτύπων. Κάθε σύναψη είναι ένα σημείο αποθήκευσης κάποιων χαρακτηριστικών της προηγούμενης εμπειρίας. Η ισχύς της σύναψης είναι μία αναπαράσταση της αποθηκευμένης γνώσης. Η λειτουργία των συνάψεων αποδίδει ένα σχετικό βάρος (σημασία) σε κάθε εισερχόμενο σήμα σύμφωνα στην προηγούμενη εμπειρία (γνώση ή μνήμη) που είναι ήδη αποθηκευμένη. Η σύναψη μαθαίνει αναπροσαρμόζοντας συνεχώς την ισχύ της (βάρος) με βάση τα νέα βιοχημικά σήματα (πληροφορίες) που εισέρχονται στον νευρώνα. Ο ρόλος του κυτταρικού σώματος είναι να εκτελεί συχνά ένα σταθμισμένο άθροισμα όλων αυτών των εισόδων, οι οποίες αξιολογούνται έτσι ώστε, αν η στάθμιση υπερβαίνει ένα ορισμένο κατώφλι, τότε ο νευρώνας θα πυροδοτήσει ανάλογες ηλεκτρικές αποκρίσεις προς την έξοδο. Ένας νευρώνας μπορεί να απεικονιστεί ως ένας επεξεργαστής πληροφοριών, ο οποίος λαμβάνει στην είσοδο ένα ν-διάστατο διάνυσμα σημάτων (inputs) και αποδίδει ένα βαθμωτό ως έξοδο, $y(t) \sim R$.

$$X(t) = [x1(t), x2(t), \ldots, xi(t), \ldots, xn(t)]^T \in R^n$$

$$X(t) \in R^n \rightarrow y(t) \in R^1$$

Με βάση την περιγραφή, μία βασική λειτουργική μονάδα ενός ANN, ένας νευρώνας (perceptron), μπορεί να παρουσιαστεί στο κάτωθι σχεδιάγραμμα :



Σχήμα 7. Τεχνητός νευρώνας (perceptron)

Αν θεωρήσουμε τον εγκέφαλο ως ένα μεγάλο, φυσικό νευρωνικό δίκτυο, οι τεχνητοί νευρώνες βασίζονται στη δομή του βιολογικού νευρώνα και χρησιμοποιούν μαθηματικές συναρτήσεις με πραγματικές τιμές για την προσομοίωση της συμπεριφοράς τους. Τέτοιοι τεχνητοί νευρώνες ονομάζονται perceptrons, μία έννοια που αναπτύχθηκε τη δεκαετία του '50 και του '60 από τον επιστήμονα Frank Rosenblatt. Λαμβάνοντας υπόψη αυτήν τη μαθηματική αναλογία, μπορούμε να αντιστοιχίσουμε τους βιολογικούς νευρώνες ως εξής:

- Δενδρίτες : x_1,x_2…. x_n ο αριθμός εισόδων που δέχεται ο νευρώνας. Μπορεί επίσης να έχουμε ως δεδομένα εισαγωγής, από έναν πίνακα n-διαστάσεων.

- Συνάψεις : w_1,w_2.... w_n τα βάρη που σχετίζονται με τους δενδρίτες. Αυτές είναι οι τιμές που αλλάζουν κατά τη διάρκεια της εκπαίδευσης, ώσπου τελικά να συμφωνήσουμε ότι ο νευρώνας είναι εκπαιδευμένος (έμαθε να εξάγει συγκεκριμένα χαρακτηριστικά).

- Πυρήνας (κυτταρικό σώμα): Λειτουργεί μία συνάρτηση, που συνδέει τις τιμές που προέρχονται από τις συνάψεις, καθορίζοντας έτσι τη συμπεριφορά του νευρώνα. Η προσομοίωση της δράσης του βιολογικού νευρώνα, επιτυγχάνεται με τη συνάρτηση

ενεργοποίησης y = φ(v), μόνο όταν υπάρχουν ορισμένα ερεθίσματα στην είσοδο που υπερβαίνουν κάποιο όριο. Έτσι ο πυρήνας συνήθως διαμορφώνεται, με μη γραμμικές συναρτήσεις.

- Άξονας. Είναι η τιμή εξόδου (Y) του νευρώνα. Μπορεί να χρησιμοποιηθεί ως είσοδος από άλλους νευρώνες.

Τα x0, θ, b είναι ο σταθερός όρος (bias or threshold) , ο οποίος έχει θεμελιώδη σημασία, διότι μας επιτρέπει να ορίσουμε το σημείο ενεργοποίησης και δεν εξαρτάται από καμία τιμή εισόδου. Αν x0 = 0 τότε w0 = 0 αγνοείται ο όρος, αν x0 = 1 τότε w0 = b και ο όρος ονομάζεται πόλωση και τέλος, αν x0 = -1 τότε w0 = θ και ονομάζεται κατώφλι.

1.2.2 Η βασική αρχιτεκτονική των νευρωνικών δικτύων

Ανάλογα το πώς είναι συνδεδεμένοι οι νευρώνες μεταξύ τους, υπάρχουν δυο βασικές κατηγορίες ANNs : α) πρόσθιας τροφοδότησης (feed forward) και β) οπίσθιας τροφοδότησης (feed backward).

Στα νευρωνικά δίκτυα πρόσθιας τροφοδότησης, οι νευρώνες είναι οργανωμένοι σε διαφορετικά επίπεδα. Οι έξοδοι των νευρώνων του ενός επιπέδου λειτουργούν ως είσοδοι στους νευρώνες του επόμενου επιπέδου, έως ότου τροφοδοτηθούν και οι μονάδες του τελευταίου επιπέδου. Δεν υπάρχει έξοδος νευρώνα ενός επιπέδου που να αποτελεί είσοδο νευρώνα του ίδιου ή προηγούμενων επιπέδου. Τέτοια ΤΝΔ είναι και τα δίκτυα οπισθοδιάδοσης (backpropagation).

Στα οπισθίως τροφοδοτούμενα δίκτυα, που καλούνται και ανατροφοδοτούμενα ΤΝΔ (recurrent ANNs), επιτρέπεται στις μονάδες ενός επιπέδου να τροφοδοτούν και μονάδες του ιδίου επιπέδου ή και προηγούμενων επιπέδων. Στα ανατροφοδοτούμενα ΤΝΔ δεν υπάρχουν συνήθως άνω του ενός ενδιάμεσα (κρυφά) επίπεδα.

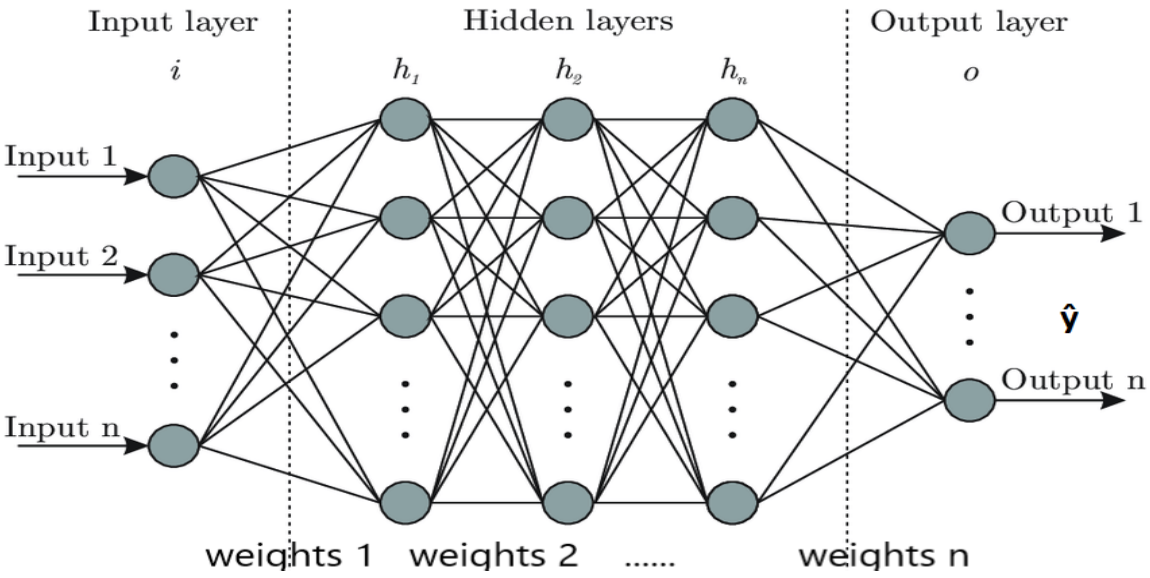
Τα νευρωνικά δίκτυα αποτελούνται από τα ακόλουθα στοιχεία:

i. Επίπεδο εισαγωγής δεδομένων (input layer).
ii. Μία αυθαίρετη ποσότητα κρυφών επιπέδων νευρώνων (hidden layers).
iii. Ένα ή πολλά επίπεδα εξόδου (output, $\hat{y}$).

iv. Ένα σύνολο βαρών και προκαταλήψεων (Weigths και bias) μεταξύ κάθε στρώματος , τα οποία μεταφέρονται μέσω των συνάψεων, των συνδέσεων δηλαδή μεταξύ των στρωμάτων.

v. Επιλογή λειτουργίας ενεργοποίησης του νευρώνα για κάθε κρυφό επίπεδο , φ(v).

Συνήθως το επίπεδο εισόδου, δεν υπολογίζεται κατά την καταμέτρηση του αριθμού των επιπέδων, σε ένα νευρωνικό δίκτυο. Αν και ένας μεμονωμένος νευρώνας μπορεί να εκτελεί ορισμένες απλές λειτουργίες ανίχνευσης προτύπων, η δύναμη υπολογισμού ενός ΤΝΔ προέρχεται από τους νευρώνες που συνδέονται σε μία δομή δικτύου. Τα μεγαλύτερα δίκτυα προσφέρουν γενικά μεγαλύτερες υπολογιστικές δυνατότητες, με τα δίκτυα πολλαπλών επιπέδων (Multi-Layer Perceptron, MLP) να είναι η πιο συχνά χρησιμοποιούμενη αρχιτεκτονική νευρωνικών δικτύων σε εφαρμογές, όπως αναγνώριση προτύπων, αναγνώριση συστήματος και έλεγχος. Οι αλγόριθμοι που χρησιμοποιούνται συχνότερα είναι αυτός της πίσω διάδοσης του λάθους (backpropagation-BP) και της κατάβασης δυναμικού (gredient decent-GD)a



Σχήμα 8. Τεχνικό Νευρωνικό Δίκτυο (ANN) πολλαπλών επιπέδων.

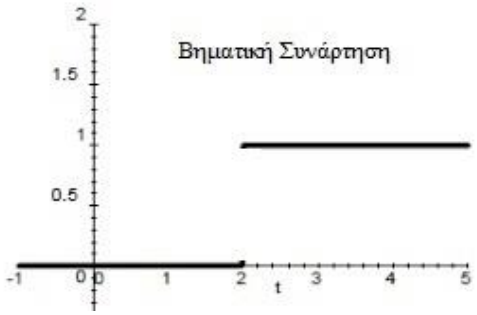
Σε ένα τυπικό νευρωνικό δίκτυο, οι νευρώνες αναφέρονται ως μονάδες επεξεργασίας (nodes) ή κύτταρα (cells) και έχουν συνάψεις - συνδέσεις, που τους επιτρέπουν να δέχονται δεδομένα (inputs) και να αποστέλλουν "σήματα" (outputs) σε άλλους νευρώνες. Ο υπολογισμός που διενεργείται σε κάθε νευρώνα έχει δύο στάδια:

i) Τον υπολογισμό του συνολικού δυναμικού των εισόδων του νευρώνα

$v(x) = \sum_{i=0}^{n} Xi * Wi$

ii) Τον υπολογισμό εξόδου y = φ(v) του νευρώνα, περνώντας το συνολικό δυναμικό από μία συνάρτηση ενεργοποίησης (activation function).

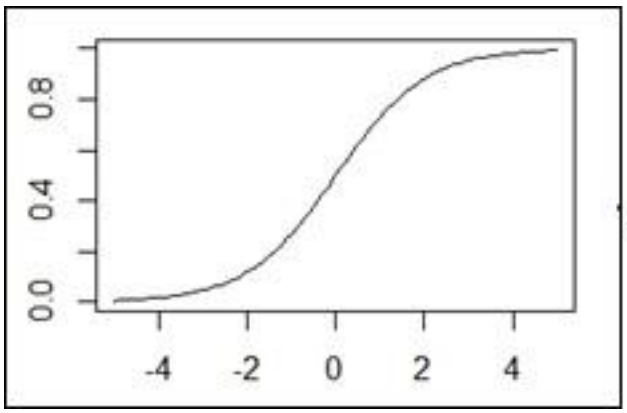
Στο πρώτο μοντέλο τεχνητού νευρώνα, ως συνάρτηση ενεργοποίησης, χρησιμοποιήθηκε η βηματική συνάρτηση, η οποία θεωρείται ότι λειτουργεί στον βιολογικό νευρώνα.



$$f(x) := \begin{cases} 1, & x > \mathrm{n} \\ 0, & x < \mathrm{n} \end{cases}$$

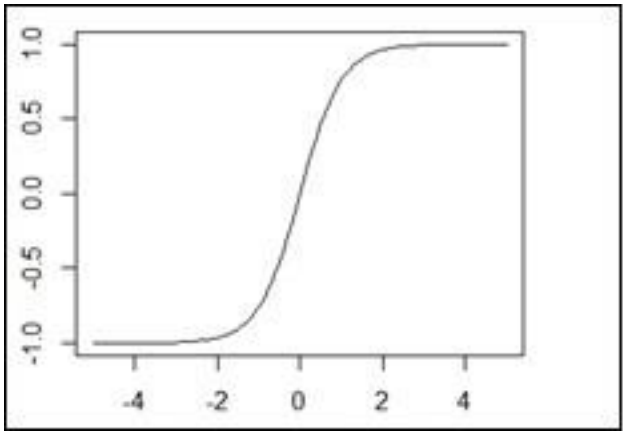
Όμως η παράγωγος της βηματικής συνάρτησης είναι μηδέν. Η μάθηση στα ANNs επιτυγχάνεται με τη ρύθμιση των βαρών και στη μαθηματική ανάλυση η μεταβολή σχετίζεται με την παράγωγο. Έτσι έχουν επιλεγεί άλλες καταλληλότερες συναρτήσεις ενεργοποίησης όπως :

Η λογιστική συνάρτηση, η οποία παρέχει τιμές στο διάστημα (0,1), με κλίση α.



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

Η υπερβολική εφαπτομένη, η οποία δίνει τιμές στο διάστημα (-1,1) με κλίση α.



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

Επίσης η συνάρτηση Gauss και η γραμμική

Gauss $f(x) = ae^{-(x-b)^2/(2c^2)}$ Γραμμική $f(x) = ax$

Τα βάρη w_{ij} αποτελούν τα στοιχεία μνήμης της αντίστοιχης διάταξης νευρώνων (i, ο νευρώνας δέκτης και j, ο νευρώνας αποστολέας). Η εκπαίδευση ενός ANN, είναι η διαδικασία ρύθμισης των συναπτικών βαρών, στη βάση ικανοποίησης κριτηρίων βελτιστοποίησης. Οι τιμές που παίρνουν τα βάρη καθορίζονται από τη διαδικασία μάθησης. Όλες οι μέθοδοι εκμάθησης κατατάσσονται σε τρεις κατηγορίες:

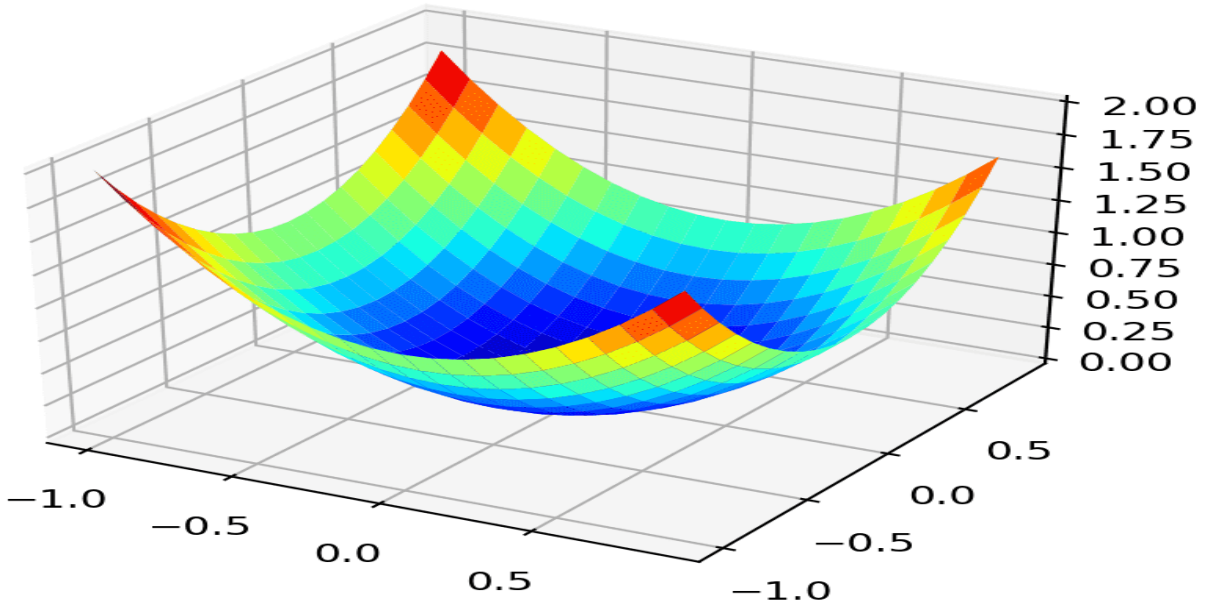
a) Μάθηση με επίβλεψη ή εποπτευόμενη μάθηση (supervised learning). Τα στοιχεία των παραδειγμάτων είναι της μορφής (είσοδος, επιθυμητή έξοδος). Είναι μία επαναληπτική τεχνική, κατά την οποία η έξοδος του ANN, συγκρίνεται με την επιθυμητή. Αν η διαφορά τους υπερβαίνει το όριο καλής λειτουργίας του δικτύου, τότε προσαρμόζουμε τα βάρη (w_i), μέχρι να το επιτύχουμε.
b) Μάθηση χωρίς επίβλεψη ή μη εποπτευόμενη μάθηση (unsupervised learning). Εδώ τα δεδομένα εκπαίδευσης, δεν περιλαμβάνουν την επιθυμητή έξοδο, αλλά μόνο τις εισόδους. Στόχος είναι η εξαγωγή κάποιων βασικών ιδιοτήτων σχετικά με τη δομή των δεδομένων εισόδου. Με αυτή τη μέθοδο γίνεται προσπάθεια επίλυσης προβλημάτων ομαδοποίησης (clustering), μείωσης της διάστασης των δεδομένων (dimensionality reduction), εκτίμηση συνάρτησης πυκνότητας πιθανότητας (pdf estimation) της στατιστικής κατανομής του συνόλου εκπαίδευσης.
c) Ενισχυμένη μάθηση ή ενισχυτική μάθηση (reinforcment learning). Εφαρμόζεται στον έλεγχο κίνησης ρομπότ, στη βελτιστοποίηση εργασιών σε εργοστάσια, στη μάθηση επιτραπέζιων παιχνιδιών, κτλ. Η έννοια της ενισχυτικής μάθησης είναι εμπνευσμένη από τα αντίστοιχα ανάλογα της μάθησης με

επιβράβευση και τιμωρίας, που συναντώνται ως μοντέλα μάθησης των έμβιων όντων. Είναι μία παραλλαγμένη τεχνική της μάθησης με επίβλεψη, αφού δεν παρέχει επιθυμητή έξοδο για κάθε είσοδο, αλλά μία τιμή που ονομάζεται σήμα ενίσχυσης και δηλώνει μόνο αν το σύστημα παρείχε σωστή ή λάθος απόκριση. Σκοπός της τεχνικής, είναι να μεγιστοποιήσει μία συνάρτηση του αριθμητικού σήματος ενίσχυσης (ανταμοιβή), για παράδειγμα την αναμενόμενη τιμή του σήματος ενίσχυσης στο επόμενο βήμα. Στο σύστημα δεν υπάρχει επίβλεψη , ούτε καθοδήγηση από κάποιον εξωτερικό παράγοντα, για το ποια ενέργεια θα πρέπει να ακολουθήσει, αλλά πρέπει να ανακαλύψει μόνο του ποιες ενέργειες είναι αυτές που θα του αποφέρουν το μεγαλύτερο κέρδος.

2.2.3 Εκπαίδευση νευρωνικών δικτύων

Η εκπαίδευση των νευρωνικών δικτύων αφορά την αναμενόμενη ή αποτελεσματική ανταπόκρισή τους, στα δεδομένα που εισάγονται στο δίκτυο. Το ζήτημα από μαθηματικής πλευράς, είναι η εύρεση των κατάλληλων παραμέτρων, μέσα από μία διαδικασία που ονομάζεται βελτιστοποίηση. Αν κοιτάξουμε πίσω την κλάση perceptron, μετράμε απλά το σφάλμα χρησιμοποιώντας τη διαφορά μεταξύ της επιθυμητής τιμής (d-Od) εξόδου και της πρόβλεψής μας (y-Oy) Όταν θέλουμε να προβλέψουμε μία συνεχή έξοδο και όχι απλώς ένα δυαδικό, θα πρέπει να χρησιμοποιήσουμε έναν διαφορετικό τρόπο για τη μέτρηση του σφάλματος, καθώς τα θετικά και τα αρνητικά σφάλματα ενδέχεται να ακυρώσουν το ένα το άλλο. Για την αποφυγή αυτού του προβλήματος η μέτρηση του σφάλματος (E), γίνεται χρησιμοποιώντας τον τύπο του τετραγωνικού σφάλματος, ο οποίος ορίζεται ως εξής: $\mathbf{E = (d-y)^2}$ και θεωρείται ο βασικός τύπος της συνάρτησης σφάλματος (ή απώλειας). Στις περισσότερες των περιπτώσεων για την συνάρτηση απώλειας χρησιμοποιείται ο τύπος του μέσου τετραγωνικού σφάλματος (MSE) : $\boldsymbol{MSE = \frac{1}{n} \sum_{\tau} (Di - Yi)^2}$

Η γραφική παράσταση της ανωτέρω συνάρτησης, είναι παραβολική καμπύλη και η βελτιστοποίηση της μέσω παραγώγισης, οδηγεί στην εύρεση του ολικού ελάχιστου (global min).



Σχήμα 9. Διαβάθμιση κλίσης (gradient descent) ενός νευρώνα με ένα βάρος, για την εύρεση ολικού ελάχιστου.

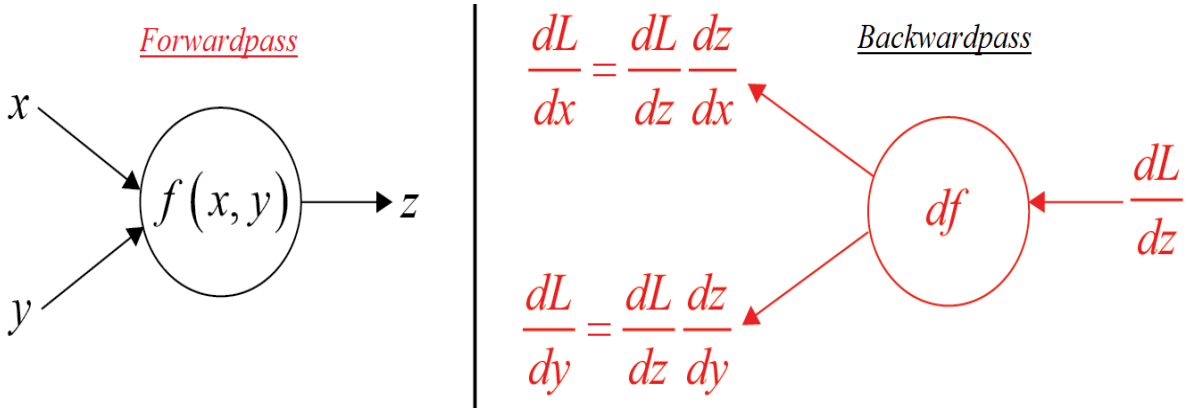
Το ολικό ελάχιστο είναι εκεί όπου έχουμε τη χαμηλότερη δυνατή απώλεια, αλλά σε πολύπλοκες λειτουργίες μπορούμε επίσης να έχουμε μερικά τοπικά ελάχιστα. Ένα τοπικό ελάχιστο ορίζεται ως το χαμηλότερο σημείο σε ένα αυθαίρετο μικρό διάστημα, οπότε δεν είναι απαραίτητα το χαμηλότερο συνολικά. Με αυτόν τον τρόπο, μπορούμε να δούμε τη διαδικασία εκπαίδευσης ως πρόβλημα βελτιστοποίησης που αναζητά αποτελεσματικά το χαμηλότερο σημείο της καμπύλης.

Ένας τρόπος για την εξερεύνηση της επιφάνειας του σφάλματος, είναι η μέθοδος διαβάθμισης κλίσης (gradient descent), η οποία χρησιμοποιεί την παράγωγο της συνάρτησης τετραγωνικού σφάλματος, σε σχέση με τα βάρη του δικτύου και ακολουθεί την κατεύθυνση προς τα κάτω. Gradient descent ονομάζουμε έναν αλγόριθμο επαναληπτικής βελτιστοποίησης πρώτης τάξης για την εύρεση ενός τοπικού ελάχιστου από μία διαφοροποιημένη συνάρτηση. Η ιδέα είναι να ακολουθήσουμε επαναλαμβανόμενα βήματα προς την αντίθετη κατεύθυνση της κλίσης (ή κατά προσέγγιση κλίσης) της συνάρτησης στο τρέχον σημείο, επειδή αυτή είναι η κατεύθυνση της απότομης καθόδου. Αντίθετα, το βήμα προς την κατεύθυνση της κλίσης θα οδηγήσει σε ένα τοπικό μέγιστο αυτής της λειτουργίας.

Η κατεύθυνση δίνεται από την κλήση με τη βοήθεια του αλγόριθμου οπισθοδιάδοσης σφάλματος (Backpropagation ή Backprop). Πρόκειται για έναν διαδεδομένο μαθησιακό αλγόριθμο που χρησιμοποιούμε σε εφαρμογές τεχνητής νοημοσύνης και είναι το μέσο με το

οποίο τα νευρωνικά δίκτυα υπολογίζουν πόσο λάθος είναι στις προβλέψεις τους. Μπορεί να θεωρηθεί ως το συμπλήρωμα του αλγόριθμου βελτιστοποίησης καθόδου κλίσης (gradient descent) που αναφέρθηκε προηγουμένως. Στον πυρήνα τους, τα νευρωνικά δίκτυα επιδιώκουν να μάθουν ένα σύνολο παραμέτρων βάρους, που τους βοηθούν να προσεγγίσουν μία λειτουργία που αναπαράγει τα εκπαιδευτικά μας δεδομένα.

Ο αλγόριθμος Backpropagation μετρά τον τρόπο με τον οποίο αλλάζει το σφάλμα μετά από κάθε κύκλο εκπαίδευσης και η κατάβαση κλίσης προσπαθεί να βελτιστοποιήσει το σφάλμα. Ο αλγόριθμος υπολογίζει την κλίση, έτσι θεωρείται ότι Backpropagation, είναι η διαδικασία υπολογισμού των διαβαθμίσεων των συναρτήσεων σφαλμάτων, μέσω της αναδρομικής εφαρμογής του κανόνα της αλυσίδας. Το σφάλμα υπολογίζεται πάντα στην έξοδο κάθε στρώματος (layer) του δικτύου και διαδίδεται ξανά μέσω του δικτύου.

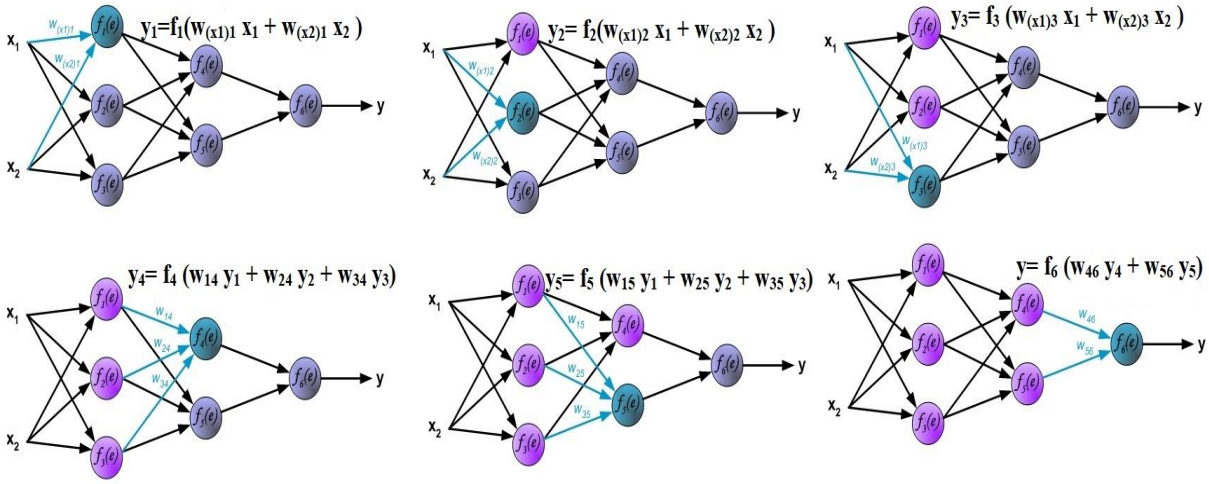


Σχήμα 10. Forwardpass - Backwardpass

Με τη μέθοδο της οπισθοδιάδοσης (Backpropagation) διοχετεύεται το σφάλμα που προέκυψε κατά το εμπρός πέρασμα (forwardpass), πίσω στα στρώματα του δικτύου μας, λαμβάνοντας τις μερικές παραγώγους της συνάρτησης απώλειας. Βασικά, αποφασίζει πόσο θα χρειαστεί ο αλγόριθμος να αλλάξει τα βάρη του για να αντισταθμίσει τυχόν κακές προβλέψεις που έκανε στο μπροστινό (forwardpass) πέρασμα. Σε κάθε κόμβο του δικτύου, υπολογίζει το σφάλμα σε σχέση με τα βάρη.

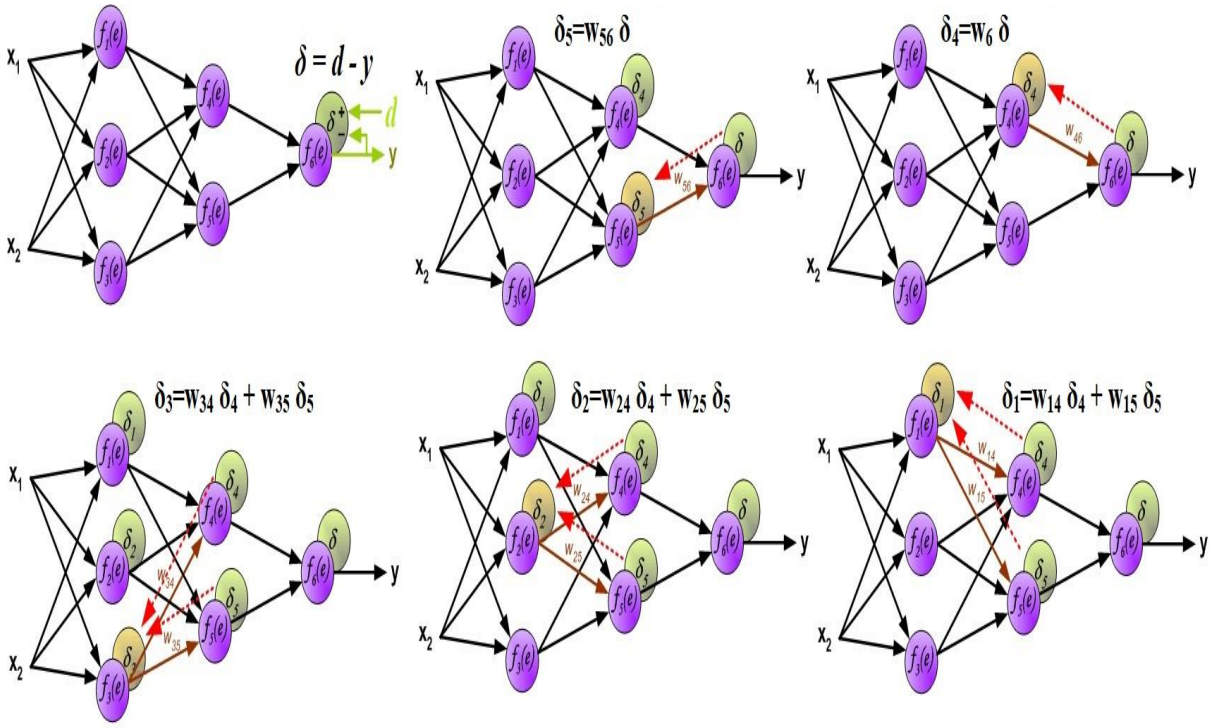
Για να "μάθει" το νευρωνικό δίκτυο, χρειαζόμαστε εκπαιδευτικό σύνολο δεδομένων, το οποίο αποτελείται από σήματα εισόδου (*X1* και *X2*) που αντιστοιχίζονται με τον αντίστοιχο στόχο (επιθυμητή έξοδος) ***d*** . Η εκπαίδευση δικτύου είναι μία επαναληπτική διαδικασία και σε κάθε επανάληψη, οι συντελεστές βαρών των κόμβων τροποποιούνται, χρησιμοποιώντας νέα δεδομένα από το σύνολο δεδομένων εκπαίδευσης. Η τροποποίηση υπολογίζεται χρησιμοποιώντας τον αλγόριθμο που περιγράφεται παρακάτω. Κάθε βήμα της εκπαίδευσης,

ξεκινά με την εξαγωγή($f_{(e)}$) και των δύο σημάτων εισόδου από το σετ προπόνησης. Μετά από αυτό το στάδιο μπορούμε να προσδιορίσουμε τις τιμές των σημάτων εξόδου για κάθε νευρώνα σε κάθε επίπεδο δικτύου. Οι παρακάτω εικόνες δείχνουν πώς διαδίδεται το σήμα μέσω του δικτύου, τα σύμβολα $W(x_i,n)$ αντιπροσωπεύουν βάρη συνδέσεων μεταξύ εισόδου x_i του νευρωνικού δικτύου και νευρώνα n στο στρώμα εισόδου. Τα σύμβολα y_n αντιπροσωπεύουν σήμα εξόδου του νευρώνα n . Οι έξοδοι σε κάθε νευρώνα του δικτύου, δύο εισόδων και μίας εξόδου, τριών επιπέδων, υπολογίζονται στο παρακάτω διάγραμμα.



Σχήμα 11. Υπολογισμός εξόδων (outputs) νευρωνικού δικτύου

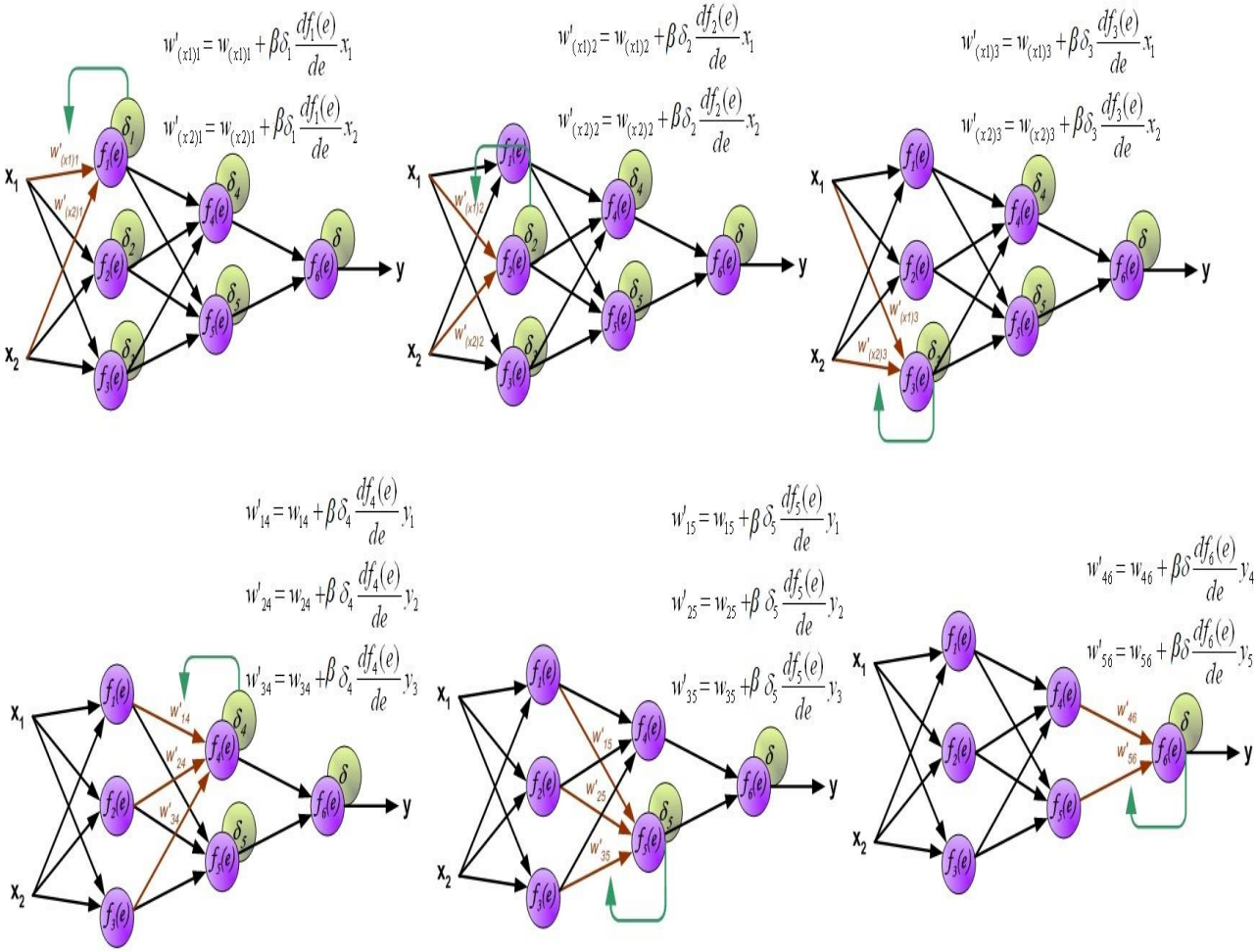
Στο επόμενο βήμα του αλγορίθμου το σήμα εξόδου (y) του δικτύου, συγκρίνεται με την επιθυμητή τιμή εξόδου d (στόχος), η οποία βρίσκεται στα ζεύγη του σύνολου δεδομένων εκπαίδευσης. Η διαφορά ονομάζεται σήμα σφάλματος δ, του νευρώνα στρώματος εξόδου. Η ιδέα είναι να μεταδοθεί το σήμα σφάλματος δ (υπολογιζόμενο σε κάθε βήμα εκπαίδευσης) σε όλους τους νευρώνες. Οι συντελεστές των βαρών W_{mn} που χρησιμοποιούνται για την οπισθοδιάδοση του λάθους, είναι ίσοι με αυτούς που χρησιμοποιήθηκαν κατά τη διάρκεια του υπολογισμού των εξόδων στην εμπρόσθια (forward pass) φάση. Μόνο η κατεύθυνση της ροής δεδομένων αλλάζει (τα σήματα μεταδίδονται αντίθετα, δηλαδή από την έξοδο στις εισόδους). Αυτή η τεχνική χρησιμοποιείται για όλα τα επίπεδα δικτύου. Εάν προκύψουν σφάλματα που προέρχονται από συσχετιζόμενους νευρώνες, προστίθενται.



Σχήμα 12. Υπολογισμός και μετάδοση σφαλμάτων (δ) σε όλους τους νευρώνες.

Μετά τον υπολογισμό του σφάλματος (δ) για κάθε νευρώνα, μπορούν να τροποποιηθούν οι συντελεστές βαρών κάθε κόμβου εισόδου νευρώνων. Στους παρακάτω τύπους η μαθηματική έκφραση df (e) / de, αντιπροσωπεύει την παράγωγο της συνάρτησης ενεργοποίησης νευρώνων (τα βάρη τροποποιούνται). Ο συντελεστής β, επηρεάζει τον ρυθμό εκπαίδευσης του δικτύου. Υπάρχουν μερικές τεχνικές για την επιλογή αυτής της παραμέτρου. Η πρώτη μέθοδος είναι να ξεκινήσει η διαδικασία διδασκαλίας με μεγάλη τιμή της παραμέτρου εκμάθησης β.

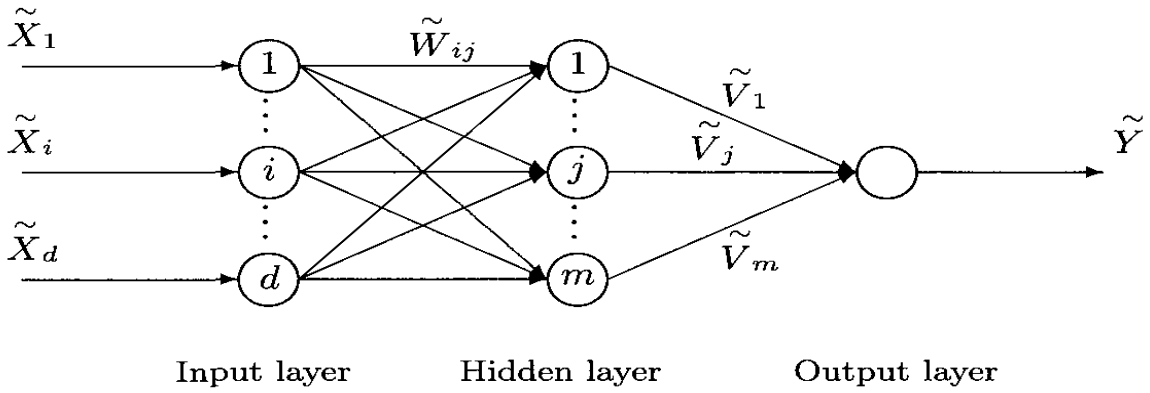
Ενώ καθορίζονται οι συντελεστές βαρών, η παράμετρος μειώνεται σταδιακά. Η δεύτερη μέθοδος ξεκινά την εκπαίδευση με μικρή τιμή παραμέτρου β. Κατά τη διάρκεια της διαδικασίας η παράμετρος αυξάνεται και στη συνέχεια μειώνεται στο τελικό στάδιο. Η εκπαίδευση του δικτύου ολοκληρώνεται θεωρητικά, όταν η τιμή του σφάλματος δ, βρεθεί κάτω από ένα καθορισμένο όριο.



Σχήμα 13. Τροποποίηση συντελεστών βαρών, με ρυθμό εκπαίδευσης, «β».

1.3 Ασαφή - Νευροασαφή Νευρωνικά Δίκτυα

Τα σημαντικά πλεονεκτήματα των ANNs, όπως η ικανότητα γενίκευσης, η ικανότητα εκμάθησης από δεδομένα εισόδου-εξόδου, η ευρωστία (robust) και τα πλεονεκτήματα της θεωρίας ασαφούς λογικής (Fuzzy Logic) , όπως η χρήση γνώσεων εμπειρογνωμόνων, εναρμονίζονται στα ασαφή νευρωνικά δίκτυα (FNN).



Σχήμα 14. Τοπολογία FNN

Τα συνήθη ANNs, παρά την ικανότητά τους να μαθαίνουν γρήγορα (εκτελώντας παράλληλη επεξεργασία των δεδομένων εισόδου) δεν μπορούν να αιτιολογήσουν ικανοποιητικά τις απαντήσεις τους. Λειτουργούν ως «μαύρα κουτιά», τα οποία δεν μπορούν να ερευνηθούν. Από τη άλλη, στα κλασικά ασαφή συστήματα μπορούμε να εξηγήσουμε επαρκώς τις απαντήσεις τους, αλλά τα συστήματα αυτά, δεν έχουν την ικανότητα μάθησης. Από την ανάγκη κάλυψης των μειονεκτημάτων και την εκμετάλλευση των πλεονεκτημάτων των δύο συστημάτων, προέκυψαν τα νευρο-ασαφή συστήματα (NFSs) σε διάφορες μορφές. Τα πλέον δημοφιλή NFSs θεωρούνται τα προσαρμοστικά νευρο-ασαφή συστήματα συμπερασμού, γνωστά ως ANFIS (Adaptive Neuro-Fuzzy Inference Systems).

Ο αλγόριθμος ANFIS αναπτύχθηκε για πρώτη φορά το 1992 από τον J.-S. Roger Jang ως ένα σύστημα ασαφών συμπερασμάτων το οποίο αποτελείται από το ασαφές μοντέλο που πρότειναν οι Takagi, Sugeno και Kang (TSK) (Takami and Sugeno, 1985), (Sugeno and Kang, 1988) για να προβάλλουν μία συστηματική προσέγγιση στη δημιουργία ασαφών κανόνων. Για μία πρώτη σειρά δύο κανόνων Sugeno fuzzy inference system, οι κανόνες μπορούν να δηλώνονται ως:

Κανόνας 1: Αν το x είναι A_1 και το y είναι B_1, τότε $f_1 = p_{1x} + q_{1y} + r_1$

Κανόνας 2: Εάν το x είναι A_2 και το y είναι B_2, τότε $f_2 = p_{2x} + q_{2y} + r_2$

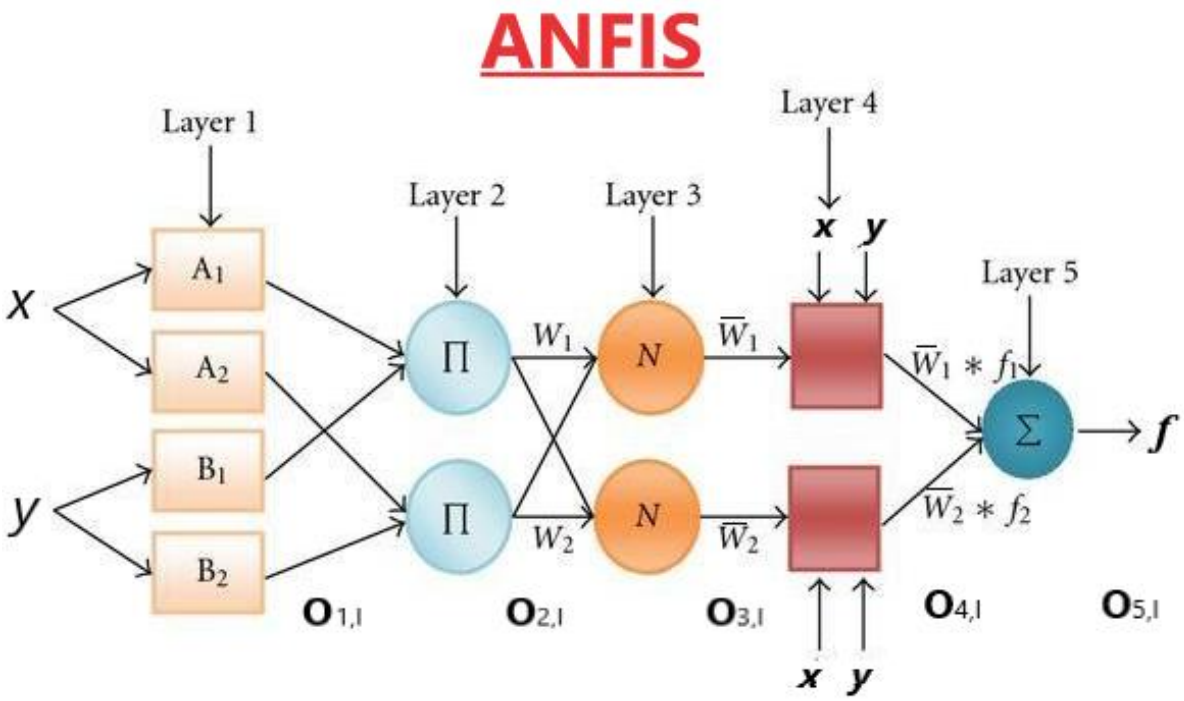
Εδώ χρησιμοποιείται το σύστημα ασαφών συμπερασμάτων τύπου 3 που προτείνουν οι Takagi και Sugeno, όπου η έξοδος κάθε κανόνα είναι ένας γραμμικός συνδυασμός των μεταβλητών εισόδου που προστίθενται με έναν σταθερό όρο. Η τελική έξοδος είναι ο σταθμισμένος μέσος όρος της παραγωγής κάθε κανόνα.

1.3.1 Δομή ANFIS

Για απλότητα, θεωρείται ότι το υπό εξέταση σύστημα ασαφών συμπερασμάτων έχει μόνο δύο εισόδους και μία έξοδο. Η βάση κανόνων περιέχει τους ασαφείς κανόνες if-then του Takagi - Sugeno - Kang ως εξής:

If x is A AND y is B, THEN z = f (x, y) ,

όπου τα A και B είναι τα ασαφή σύνολα και το z = f (x, y) είναι συνήθως ένα πολυώνυμο για τις μεταβλητές εισόδου x και y. Αλλά μπορεί επίσης να είναι οποιαδήποτε άλλη λειτουργία που μπορεί να περιγράψει την έξοδο του συστήματος εντός της ασαφούς περιοχής όπως καθορίζεται. Όταν το f (x, y) είναι μία σταθερά, σχηματίζεται ένα μηδενικό μοντέλο Sugeno fuzzy, το οποίο μπορεί να θεωρηθεί ότι είναι μία ειδική περίπτωση συστήματος Mamdani fuzzy inference.



Σχήμα 15: Δομή ANFIS

Έχοντας υπόψη, ότι οι κύκλοι απεικονίζουν σταθερούς κόμβους, ενώ τα τετράγωνα προσαρμοζόμενους κόμβους, τα 5 επίπεδα (Layers) του ANFIS περιγράφονται κάτωθι:

Layer 1 : $O_{1,i}$ είναι η έξοδος του i-οστού κόμβου του επιπέδου 1. Κάθε κόμβος i σε αυτό το επίπεδο είναι ένας προσαρμοστικός κόμβος με μία λειτουργία.

$$O_{1,i} = \mu A_i(x) \text{ for } i = 1,2, \text{ for } \quad O_{1,i} = \mu B_{i-2}(x) \text{ for } i = 3,4$$

(χ, y) είναι το επίπεδο εισόδου και Ai (ή B_{i-2}) είναι μία γλωσσική ετικέτα (μικρή, μεγάλη) που σχετίζεται με αυτόν τον κόμβο. Επομένως, το $O_{1,i}$ είναι ο βαθμός συμμετοχής ενός ασαφούς συνόλου (A_1,A_2, B_1,B_2). Οι είσοδοι x , y εισέρχονται στη συνάρτηση συμμετοχής (bell) στον κόμβο i. $\mu A(x) = \frac{1}{1+\left|\frac{x-ci}{ai}\right|^{2bi}}$ (i,δείκτης,# κόμβου)

Όπου αi bi, ci είναι το σύνολο παραμέτρων της συνάρτησης συμμετοχής. Σε αυτό το επίπεδο κάθε κόμβος περιέχει μία συνάρτηση συμμετοχής.

Layer 2 : Ο κόμβος i είναι σταθερός, επισημαίνεται με Π και η έξοδος είναι το γινόμενο των εισερχόμενων συναρτήσεων συμμετοχής. Εξετάζεται αν οι μεταβλητές εισόδου, ικανοποιούν από κοινού τον κανόνα και λαμβάνεται η τομή των συναρτήσεων συμμετοχής. Η έξοδος του κάθε κόμβου αντιπροσωπεύει τη δύναμη ενός κανόνα. Η μορφή της συνάρτησης εξόδου για τον κάθε κανόνα, είναι η εξής : $O_{2,i} = w_i = \mu A_i(x) \bullet \mu B_i(y), \quad i = 1,2$

Κάθε κόμβος αντιπροσωπεύει τη δύναμη πυροδότησης του κανόνα.

Layer 3 : Εδώ έχουμε σταθερούς κόμβους με ετικέτα N και σε αυτό το επίπεδο, ο i-οστός κόμβος υπολογίζει την αναλογία της ισχύος πυροδότησης του i-οστού κανόνα, προς το άθροισμα όλων των δυνατοτήτων πυροδότησης του κανόνα. $O_{3,i} = \overline{\overline{w_i}} = \frac{wi}{w1+w2}$, $i=1,2$

Οι έξοδοι είναι τα κανονικοποιημένα ασαφή βάρη.

Layer 4 : Κάθε κόμβος i σε αυτό το επίπεδο είναι ένας προσαρμοστικός κόμβος με συνάρτηση κόμβου: $O_{4,i} = \overline{w}_i f_i = \overline{w}_i (p_i x + q_i y + r_i)$

$\overline{\overline{w_i}}$ είναι ο κανονικοποιημένος βαθμός ενεργοποίησης από το επίπεδο 3.

pi, qi, ri, είναι το σύνολο παραμέτρων αυτού του κόμβου και αναφέρονται ως επακόλουθες παράμετροι. Η συνάρτηση είναι γραμμική ή σημείο και είναι προκαθορισμένη στο ANFIS.

Layer 5 : Ο μοναδικός κόμβος σε αυτό το επίπεδο είναι ένας σταθερός κόμβος αθροιστής, ο οποίος υπολογίζει τη συνολική έξοδο, ως το άθροισμα όλων των εισερχόμενων σημάτων: Συνολική Έξοδος $= O_{5,1} = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f i}{\sum_i w_i}$

Το ANFIS μπορεί να εκπαιδευτεί από έναν υβριδικό αλγόριθμο μάθησης που παρουσίασε ο Jang.

- Στο πέρασμα προς τα εμπρός (“forward pass”), ο αλγόριθμος χρησιμοποιεί τη μέθοδο των ελαχίστων τετραγώνων για να προσδιορίσει τις επακόλουθες παραμέτρους στο επίπεδο 4.
- Στο πέρασμα προς τα πίσω, τα σφάλματα διαδίδονται προς τα πίσω και οι παράμετροι της αρχής ενημερώνονται κατά κλίση.

1.3.2 Παράδειγμα κατανόησης της δομής ANFIS

Η διατύπωση δύο κανόνων με βάση το Sugeno fuzzy inference system είναι:

Κανόνας 1 : If x is small (A_1) AND y is small (B_1) THEN f1 = small

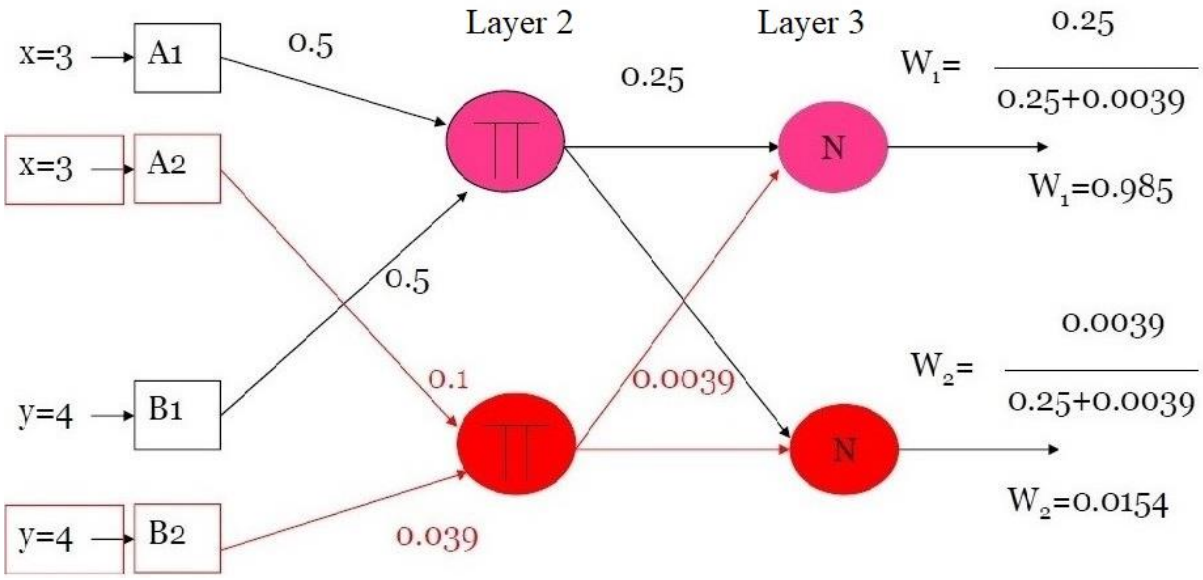
Κανόνας 2 : If x is large (A2) AND y is large (B2) THEN f2 = large

(p1, q1, r1) = (0.1, 0.1, 0.1), (p2, q2, r2) = (10, 10, 10), Bell MFs

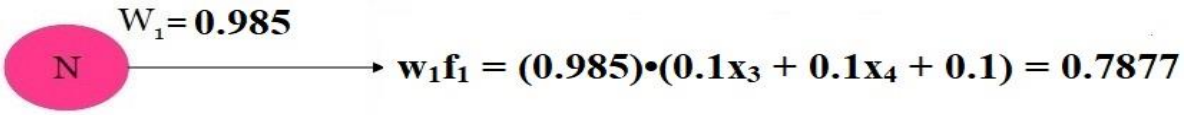
Layer 1

A1: $\mu_{A1}(x) = \frac{1}{1+\left|\frac{x-1}{2}\right|^2}$ B1: $\mu_{B1}(y) = \frac{1}{1+\left|\frac{y-2}{2}\right|^2}$ $f1 = 0.1x + 0.1y + 0.1$

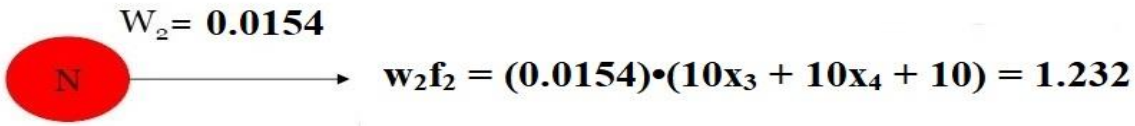
A2: $\mu_{A2}(x) = \frac{1}{1+\left|\frac{x-9}{2}\right|^2}$ B2: $\mu_{B2}(y) = \frac{1}{1+\left|\frac{y-14}{2}\right|^2}$ $f2 = 10x + 10y + 10$

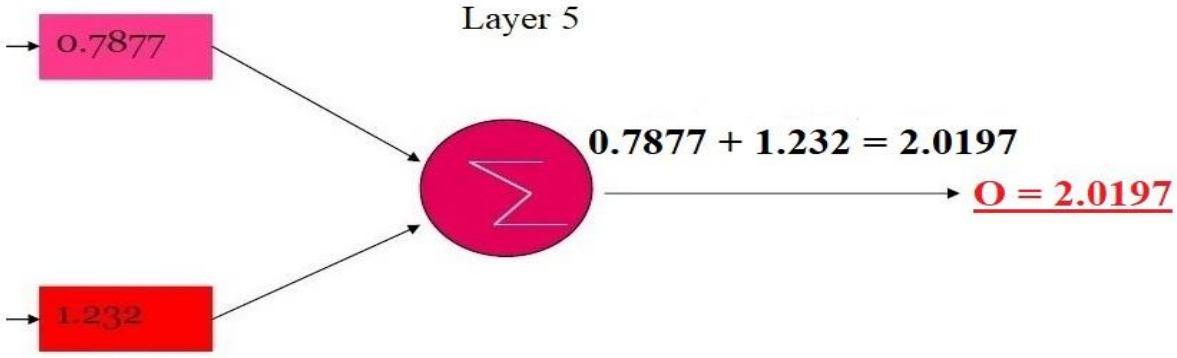


Οι τιμές των βαρών W_1, W_2, περνούν στο τελευταίο στρώμα πριν την έξοδο.



Layer 4





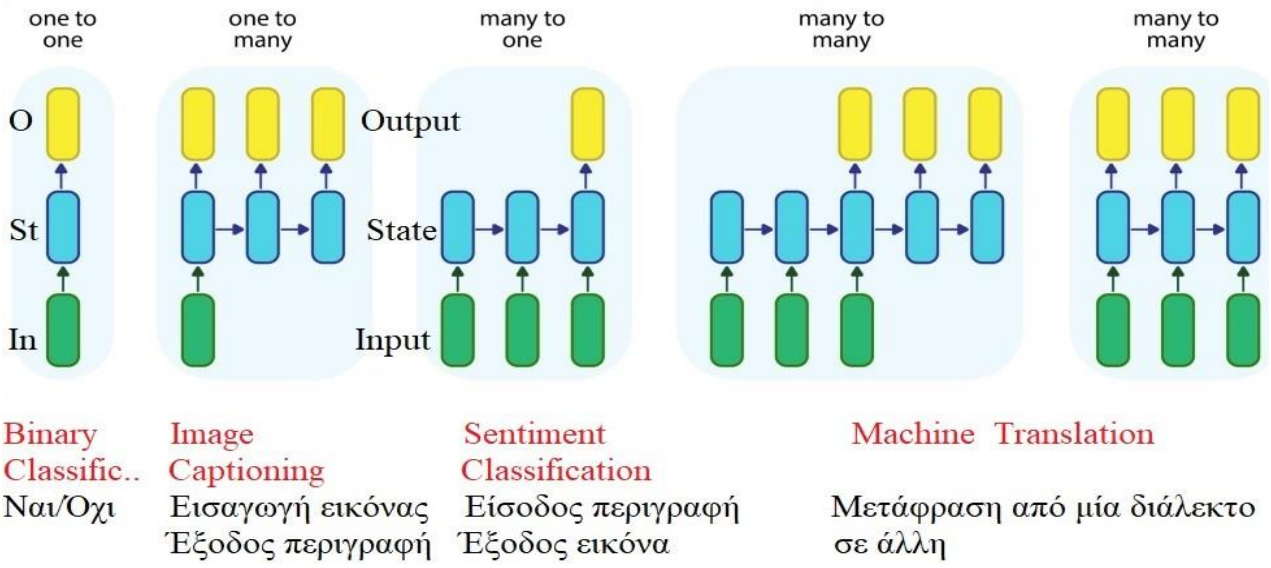
1.3.3 Αξιολόγηση ANFIS

Το νευροασαφές σύστημα ANFIS θεωρείται μία εύκολη εφαρμογή, για γρήγορη και ακριβή εκμάθηση του δικτύου με μεγάλες ικανότητες γενίκευσης και επαρκή αιτιολόγηση. Κάνοντας χρήση ασαφών κανόνων διευκολύνεται η ενσωμάτωση γλωσσικής και αριθμητικής γνώσης για την επίλυση προβλημάτων.

Οι κυριότεροι περιορισμοί του ANFIS είναι ότι βασίζεται σε ένα Fuzzy Inference System τύπου Sugeno, έχει μία έξοδο που λαμβάνεται με τη μέθοδο της από-ασαφοποίησης σταθμισμένου μέσου κι όλες οι συναρτήσεις συμμετοχής εξόδου πρέπει να είναι ίδιου τύπου, είτε γραμμικές, είτε σταθερές. Δεν μπορεί να γίνεται κοινή χρήση κανόνων. Αυτό σημαίνει ότι διαφορετικοί κανόνες δεν μπορούν να έχουν την ίδια συνάρτηση συμμετοχής (MF) στην έξοδο, επίσης δεν μπορεί να χρησιμοποιηθεί οποιαδήποτε συνάρτηση συμμετοχής, παρά μόνο προκαθορισμένες που επιβάλλουν οι ορισμοί του ANFIS. Ο κάθε κανόνας πρέπει να είναι αυτοτελής και να υπάρχουν βάρη σε κάθε κανόνα.

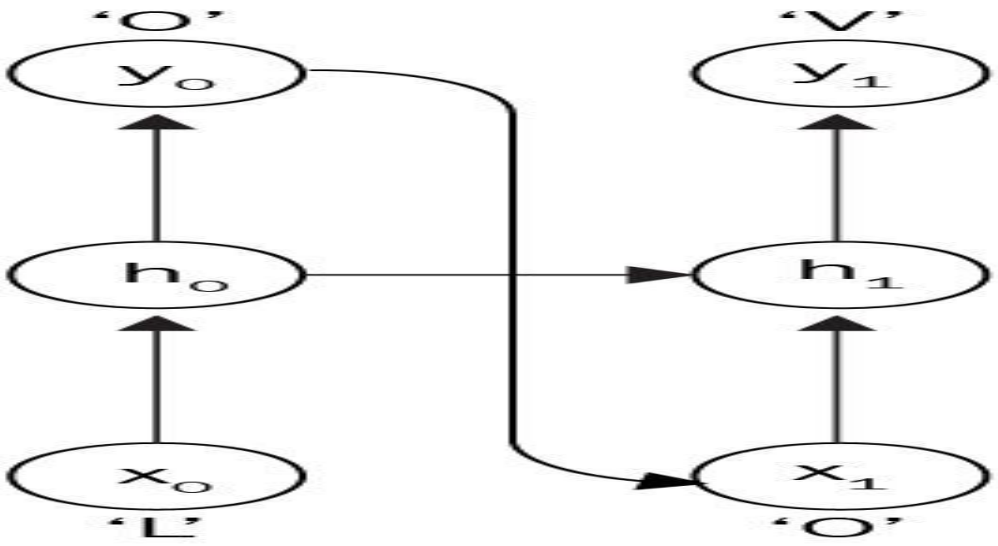
1.4 Αναδρομικά Νευρωνικά Δίκτυα (Recurrent Neural Networks – RNNs)

Τα RNNs παρουσιάστηκαν για πρώτη φορά στη δεκαετία του 1980 με την εφεύρεση του δικτύου Hopfield. Αργότερα το 1997, οι Hochreiter και Schmidhuber πρότειναν ένα προηγμένο μοντέλο RNN που ονομάζεται μακροπρόθεσμη μνήμη (LSTM - Long-Sort Term Memory). Μία πιο πρόσφατη βελτίωση στην οικογένεια RNN παρουσιάστηκε το 2014 από τους Chung et al. Αυτή η νέα αρχιτεκτονική, που ονομάζεται Gated Recurrent Unit (GRU) λύνει το ίδιο πρόβλημα με το LSTM αλλά με απλούστερο τρόπο. Τα RNNs χρησιμοποιούνται ευρέως για περιπτώσεις που περιλαμβάνουν διαδοχικά δεδομένα, όπως χρονοσειρές, κείμενο, ήχο, ομιλία, βίντεο, η οπτική πληροφορία που προκύπτει από μία κίνηση ή μία δράση, καιρός και πολλά άλλα. Έχουν χρησιμοποιηθεί αρκετά σε διάφορες εργασίες φυσικής γλώσσας (NLP), όπως μετάφραση διαλέκτων, ανάλυση συναισθημάτων, δημιουργία κειμένου και ούτω καθεξής.



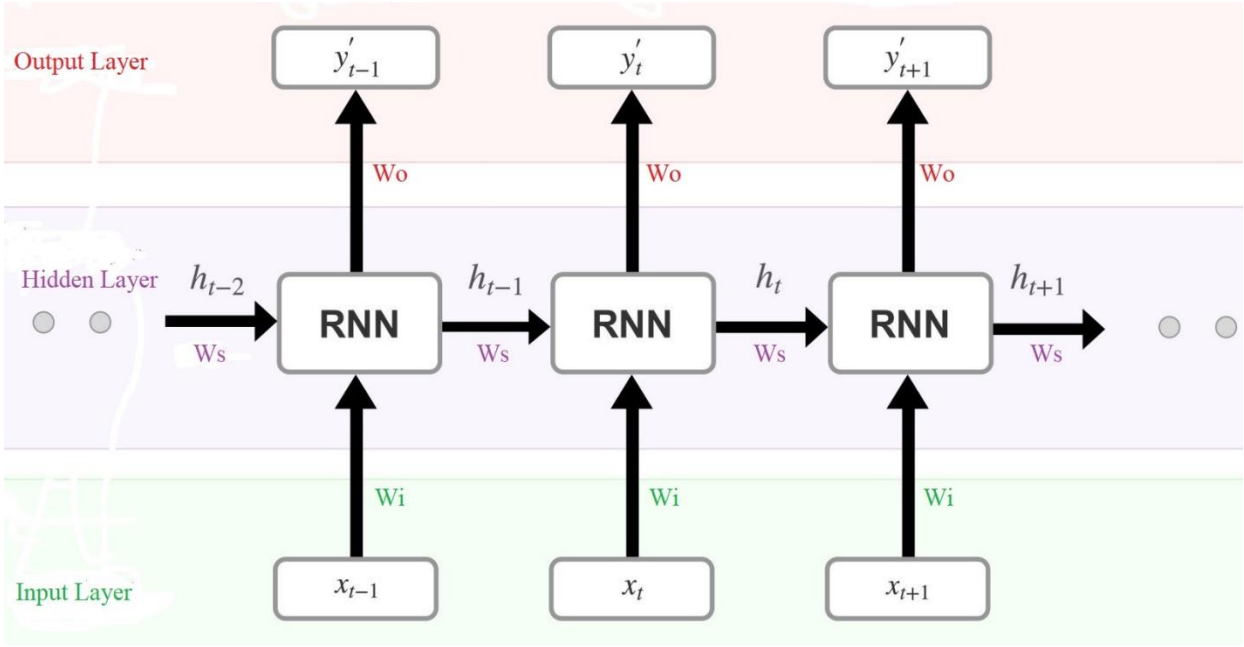
Σχήμα 16 : Εφαρμογές RNNs

Τα αναδρομικά δίκτυα, προσπαθούν να συλλάβουν πληροφορίες από προηγούμενες εισόδους διατηρώντας στοιχεία εσωτερικής μνήμης. Αν θέλουμε να μεταφράσουμε μία συγκεκριμένη λέξη σε μία πρόταση, δεν μπορούμε να χρησιμοποιήσουμε παραδοσιακά FNN για αυτό, επειδή δεν θα μπορούν να χρησιμοποιήσουν τη μετάφραση των προηγούμενων λέξεων ως εισαγωγή με την τρέχουσα λέξη που θέλουμε να μεταφράσουμε. Αυτό μπορεί να οδηγήσει σε εσφαλμένη μετάφραση λόγω της έλλειψης σχετικών πληροφοριών γύρω από αυτήν τη λέξη. Τα RNN διατηρούν πληροφορίες σχετικά με το παρελθόν και έχουν κάποιο είδος βρόχου για να επιτρέψουν τη χρήση των πληροφοριών που έχουν ήδη μάθει για την τρέχουσα πρόβλεψη σε οποιοδήποτε δεδομένο σημείο.



Σχήμα 17 : Ροή δεδομένων σε RNN

Στα RNNs, ο βρόχος ανατροφοδότησης μεταβιβάζει τις πληροφορίες της τρέχουσας κατάστασης h_0 στην επόμενη h_1. Η κατάσταση του δικτύου σε οποιοδήποτε t μπορεί να γραφτεί ως $h_t = f(h_{t-1}, x_t)$. Η διεργασία *f*, εκτελείται σε κάθε στοιχείο της ακολουθίας και μία οποιαδήποτε κατάσταση έστω h_t , εξαρτάται από την προηγούμενη κατάσταση h_{t-1} με βάση προηγούμενους υπολογισμούς και τις εισόδους της h_t . Σε αυτήν την αρχιτεκτονική που μοιάζει με αλυσίδα, ή την πρόσθετη καταγραφή μνήμης αυτού που έχει υπολογιστεί μέχρι στιγμής, οφείλεται η μεγάλη επιτυχία στην εφαρμογή RNNs σε δεδομένα χρονοσειρών και ακολουθιών.



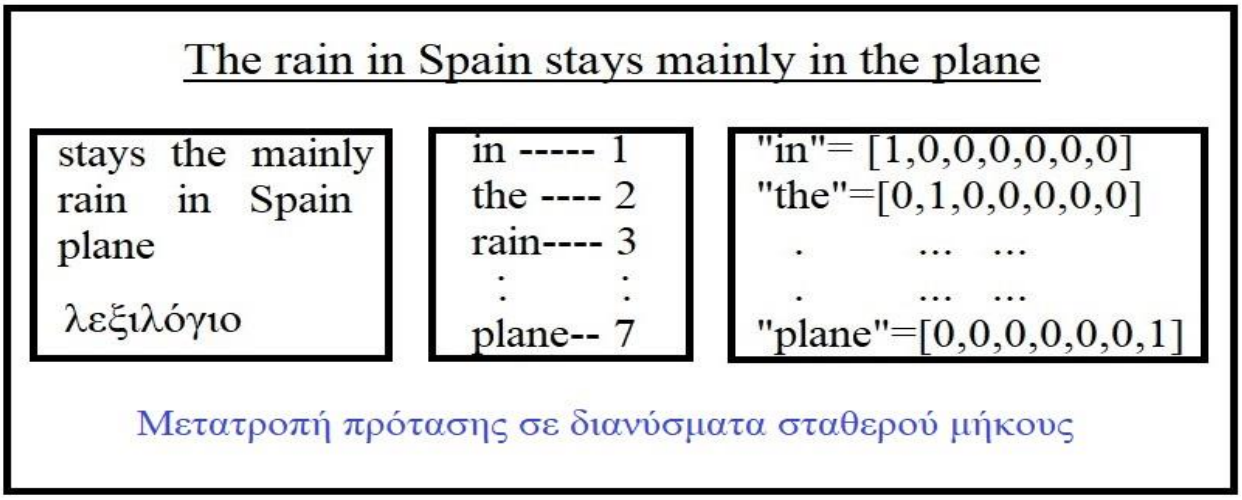
Σχήμα 18 : Αρχιτεκτονική RNN με τα βάρη (W) που πρέπει να εκπαιδευτούν

Με *Wi* συμβολίζεται η μήτρα των βαρών των εισόδων (*X*i), της κατάστασης (h_i). Με *Wo* αναγνωρίζεται η μήτρα των βαρών των εξόδων (y'_i), ενώ το σύμβολο *Ws* αντιστοιχεί στη μήτρα βαρών μεταφοράς της κατάστασης h_i στην επόμενη χρονικά κατάσταση h_{i+1} . Η συνάρτηση ενεργοποίησης φ(•) του νευρώνα είναι μη γραμμική και συνήθως εφαρμόζεται η σιγμοειδής ή tahn ή ReLu ή άλλη, στον αναδρομικό τύπο :

$$h_i = \varphi(Xi * Wi + h_{i-1} * Ws) \quad (1)$$

Η είσοδος *X*i κάθε νευρώνα, πολλαπλασιάζεται με τον πίνακα βαρών των εισόδων (*X*i * Wi). Το αποτέλεσμα προστίθεται στο γινόμενο της προηγούμενης κατάστασης (h_{i-1}), με τον πίνακα βαρών μεταφοράς (h_{i-1} * Ws). Το αποτέλεσμα του αθροίσματος, περνάει μέσα από τη συνάρτηση ενεργοποίησης φ(•) του νευρώνα, όπου παράγεται η τιμή της h_i κατάστασης.

Τέλος ο φορέας εξόδου υπολογίζεται, λαμβάνοντας τον γραμμικό συνδυασμό των καταστάσεων (h_i) με την αντίστοιχη μήτρα βάρους τους *Wo*. Η τιμή της h_i (μήτρα ή αριθμός) πολλαπλασιάζεται με τη μήτρα των εξόδων Wo, για να πάρουμε την έξοδο του νευρώνα, $y_i = h_i * Wo$. Ανάλογα με την εφαρμογή, η έξοδος θα μπορούσε επίσης να υπολογιστεί σε μία λειτουργία softmax : y_i = softmax ($h_i * Wo$). Αυτή ακριβώς η έξοδος χρησιμοποιείται ως είσοδος X_{i+1} στον νευρώνα της επόμενης χρονικής κατάστασης και πολλαπλασιάζεται με τη μήτρα βαρών εισόδου *Wi* , για να ακολουθηθεί η ίδια διαδικασία που θα παράγει την επόμενη χρονικά έξοδο y_{i+1} . Οι έξοδοι δεν εξαρτώνται μόνο από τις τρέχουσες εισόδους, αλλά και από προηγούμενους υπολογισμούς, την προηγούμενη χρονικά γνώση, η οποία μοιάζει να αποθηκεύεται σε κάθε προγενέστερη κατάσταση. Τα νευρωνικά δίκτυα λειτουργούν εκτελώντας μαθηματικούς υπολογισμούς, με βάση αριθμητικά δεδομένα τα οποία εισάγονται στο δίκτυο, επεξεργάζονται και παράγουν αριθμητικές εξόδους. Αυτό επιτυγχάνεται με τον μετασχηματισμό των λεκτικών εισόδων, σε διανύσματα σταθερού μήκους. Αν για παράδειγμα θέλαμε να εκπαιδεύσουμε ένα RNN, να προβλέπει τις λέξεις της πρότασης, “The rain in Spain stays mainly in the plane”, θα ήταν σχεδόν αδύνατο να γίνει με λεκτικές εισόδους. Έτσι διαχωρίζουμε και μετατρέπουμε τις λέξεις της πρότασης σε διανύσματα σταθερού μήκους ως κάτωθι:



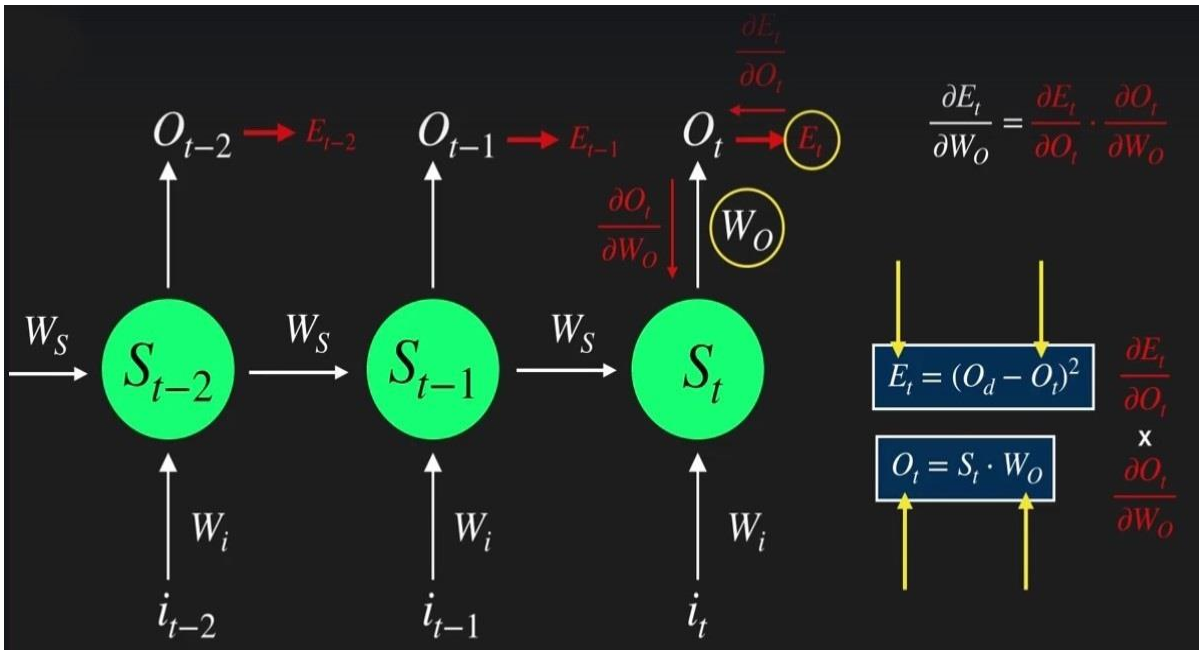
Σχήμα 19. Παράδειγμα μετατροπής λεκτικών εισόδων σε εισόδους σταθερών διανυσμάτων για RNN.

1.4.1 Εκπαίδευση RNNs

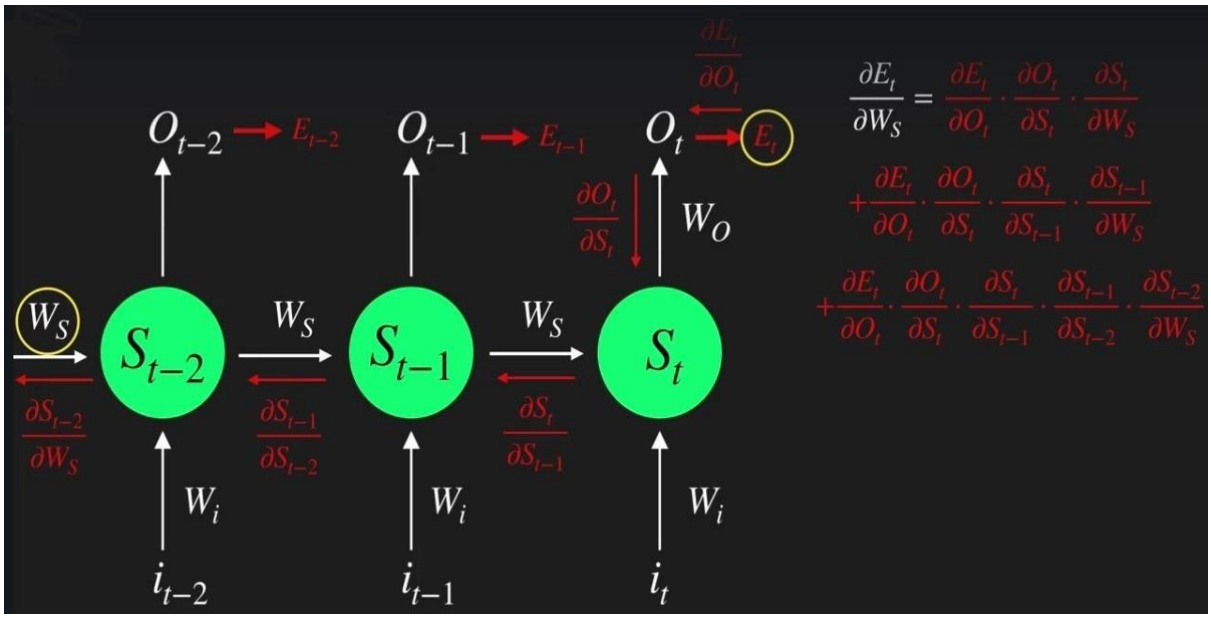
Η μέθοδος εκπαίδευσης των RNNs, κατατάσσεται στην κατηγορία της εποπτευόμενης μάθησης, όπου εισάγουμε συγκεκριμένα ζεύγη εισόδων(W_i)-εξόδων(O_d). Κάθε εκτιμώμενη έξοδος O_i , έχει μία απόκλιση από την πραγματική ή επιθυμητή O_d, η οποία υπολογίζεται με τη συνάρτηση απώλειας (Ei -loss function) για κάθε χρονικό βήμα χωριστά. Η συνάρτηση που χρησιμοποιείται κατά κόρον είναι αυτή για τον υπολογισμό του μέσου τετραγωνικού σφάλματος (MSE). Η απόκλιση του χρονικού βήματος t, υπολογίζεται από τον τύπο :

$$\mathbf{Et = (Od - Ot)^2}$$

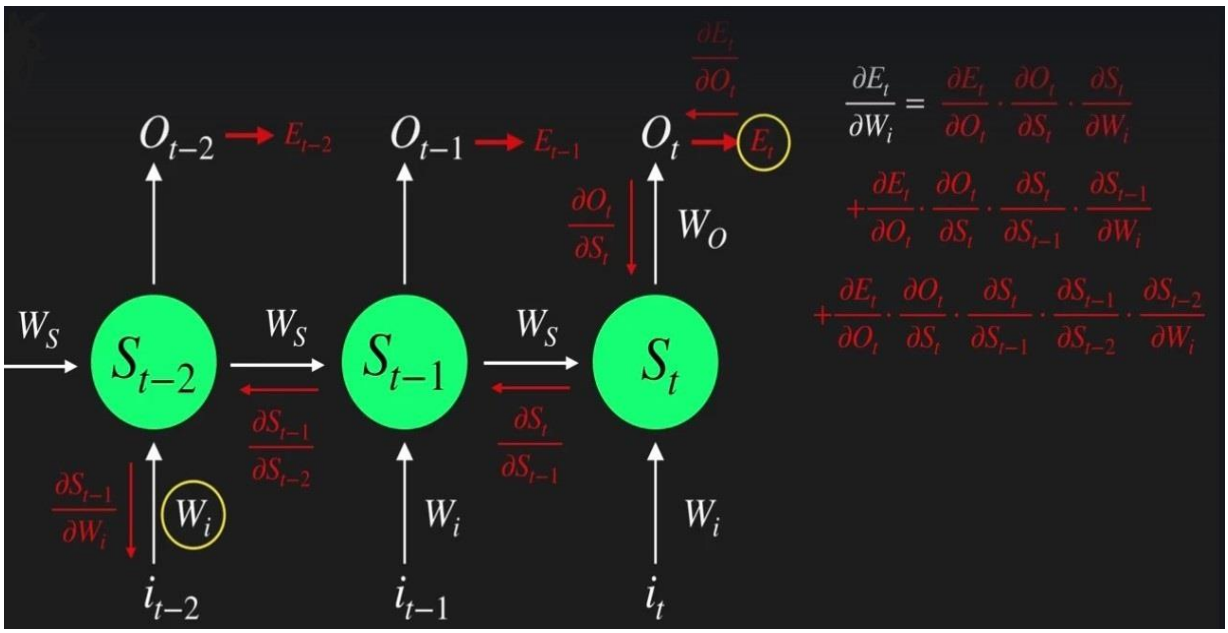
Εμείς προσθέτουμε τις απώλειες (Ei) κάθε βήματος (i), για να καθορίσουμε την ολική απώλεια (total loss), η οποία θα μας φανεί χρήσιμη στην εκπαίδευση του δικτύου. Οι μήτρες των βαρών (W), παραμένουν ίδιες σε κάθε χρονικό βήμα, σε κάθε δηλαδή κατάσταση, κατά το εμπρός πέρασμα της διαδικασίας. Για τη βελτιστοποίηση της συνάρτησης απώλειας, τα βάρη αναπροσαρμόζονται κατά τη διαδικασία της εκπαίδευσης, με τον αλγόριθμο Backpropagation through time.



Σχήμα 20. Ελαχιστοποίηση της συνάρτησης απώλειας, ως προς τα βάρη εξόδων (*Wo*) του δικτύου.



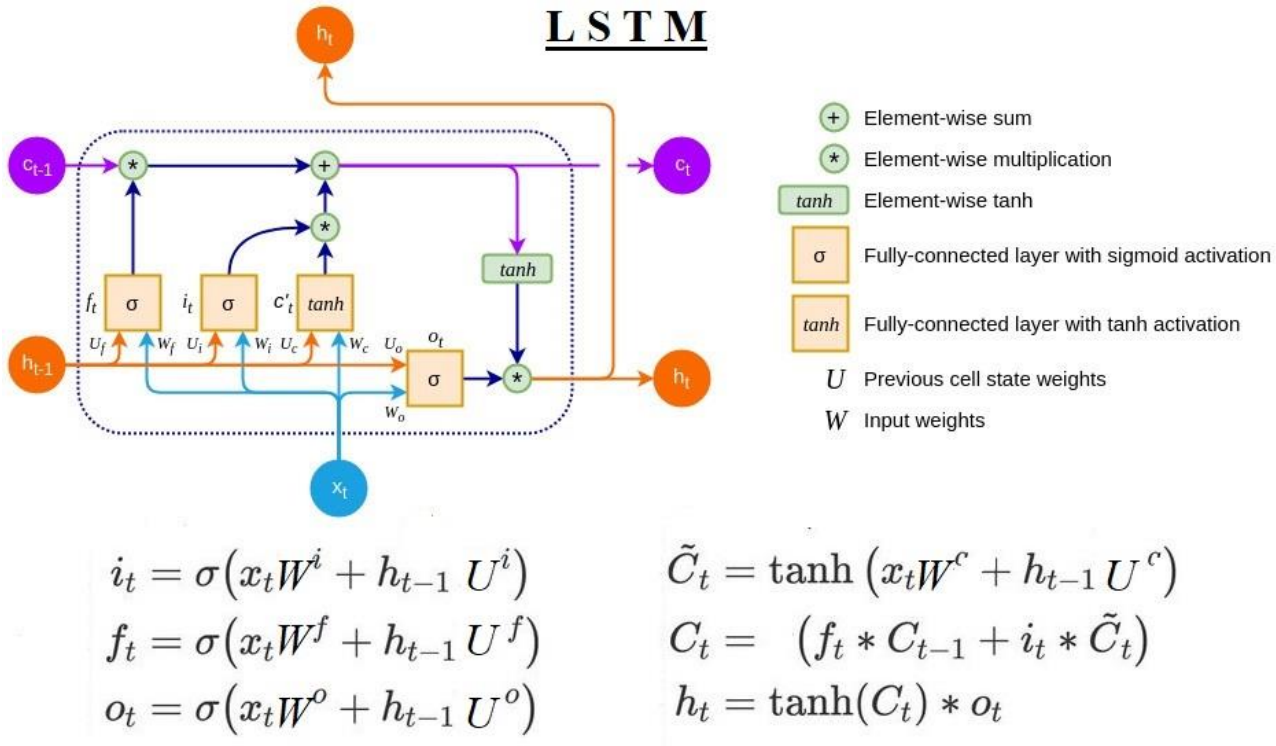
Σχήμα 21. Ελαχιστοποίηση της συνάρτησης απώλειας, ως προς τα βάρη κατάστασης (*Ws*) του δικτύου.



Σχήμα 22. Ελαχιστοποίηση της συνάρτησης απώλειας, ως προς τα βάρη εισόδων (*Wi*) του δικτύου.

1.4.2 LSTM

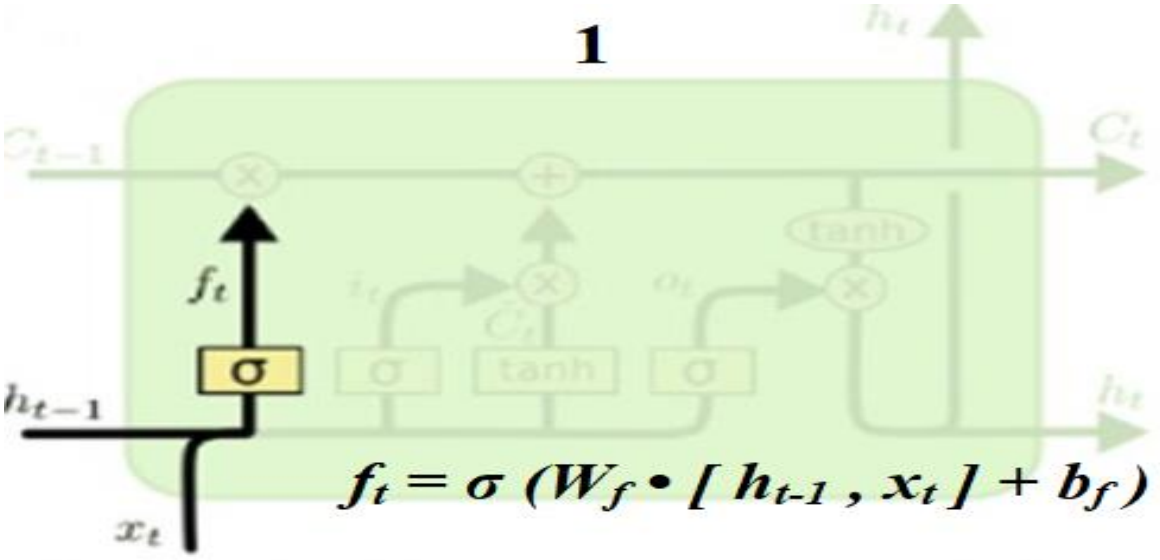
Ο νευρώνας LSTM έχει τη δυνατότητα να αφαιρεί ή να προσθέτει πληροφορίες μέσα από δομές που ονομάζονται πύλες. Οι πύλες i , f , και o, αντιπροσωπεύουν τις πύλες εισόδου, λήθης και εξόδου αντίστοιχα, και διαμορφώνονται από στρώματα (layers), που διαπερνούν σιγμοειδείς συναρτήσεις. Οι έξοδοι παράγουν αριθμούς από το μηδέν (0) έως το ένα (1), ελέγχοντας πόση πληροφορία από αυτές τις πύλες πρέπει να περάσει. Η πύλη εισόδου μπορεί συνήθως να επιτρέπει ή να απορρίπτει εισερχόμενα σήματα ή εισόδους, για να αλλάξει τη μνήμη στην κατάσταση του νευρώνα (cell). Η πύλη εξόδου μεταδίδει συνήθως την τιμή (output) σε άλλους νευρώνες. Η πύλη της λήθης, ελέγχει τη μνήμη της αναδρομικής σύνδεσης του νευρώνα, ώστε να μπορεί να θυμάται ή να ξεχνά τις προηγούμενες καταστάσεις. Πολλαπλά LSTM συνήθως στοιβάζονται σε οποιοδήποτε δίκτυο βαθιάς μάθησης για την επίλυση πραγματικών προβλημάτων.



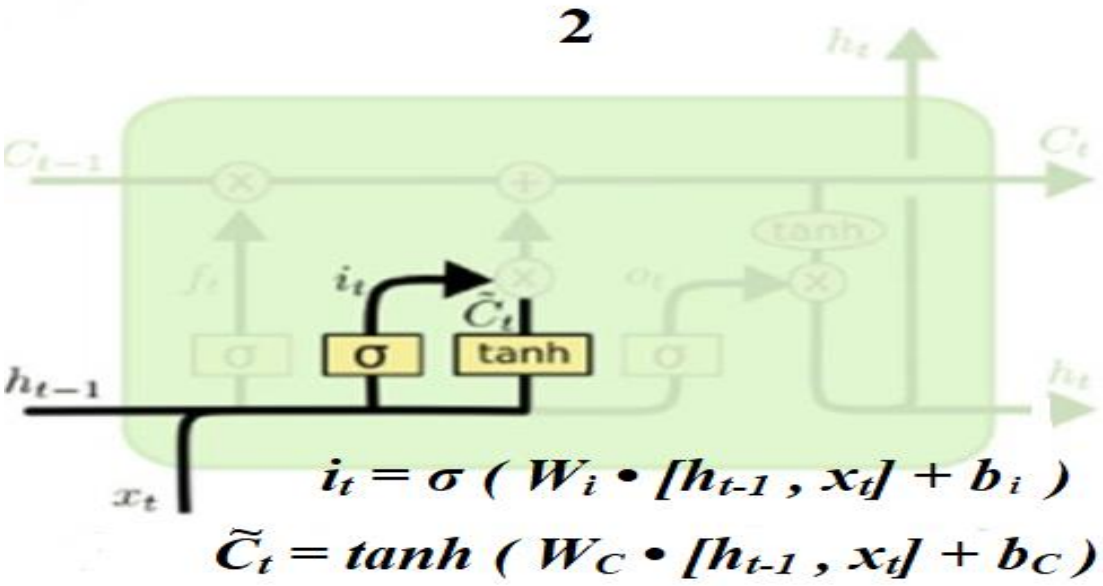
$$i_t = \sigma\left(x_t W^i + h_{t-1}\, U^i\right) \qquad \tilde{C}_t = \tanh\left(x_t W^c + h_{t-1}\, U^c\right)$$
$$f_t = \sigma\left(x_t W^f + h_{t-1}\, U^f\right) \qquad C_t = \left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$o_t = \sigma\left(x_t W^o + h_{t-1}\, U^o\right) \qquad h_t = \tanh(C_t) * o_t$$

Σχήμα 23. Αρχιτεκτονική LSTM, ροή πληροφοριών και μαθηματικές διεργασίες εντός του νευρώνα.

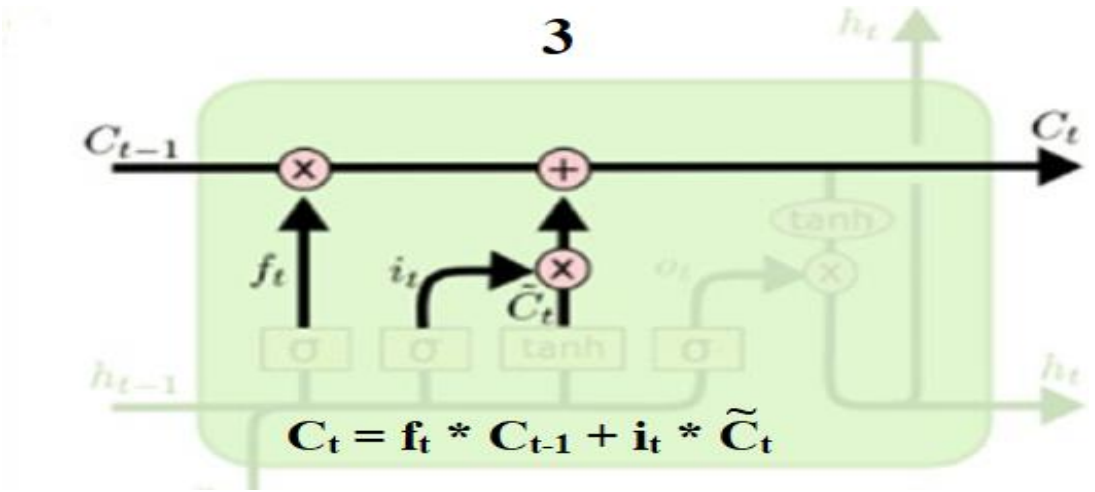
Μία λεπτομερής αρχιτεκτονική ενός κελιού LSTM με τη ροή των πληροφοριών, παρουσιάζεται στα ακόλουθα διαγράμματα.



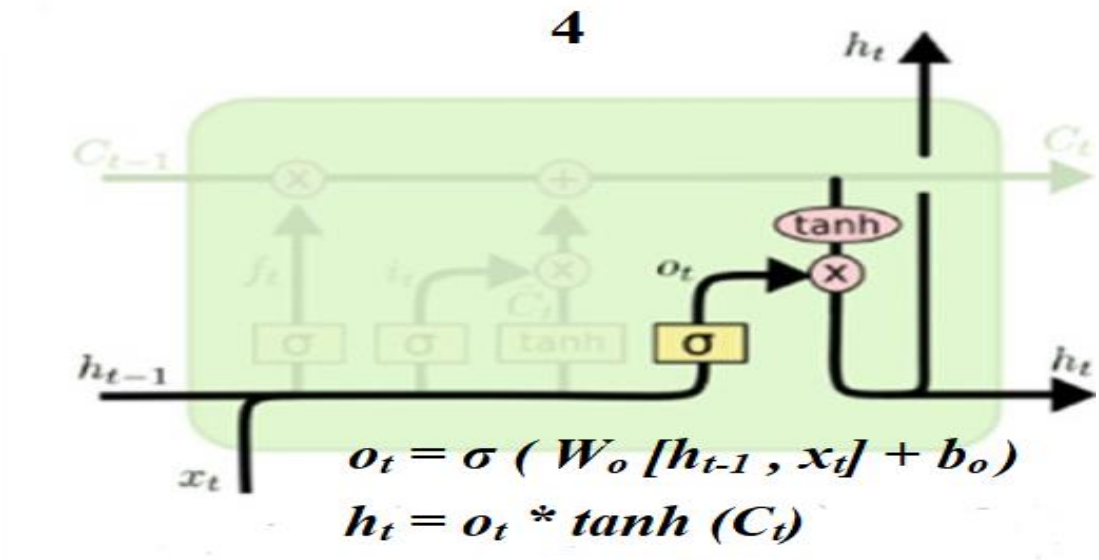
1. Αποφασίζει ποιες πληροφορίες θα αφαιρεθούν από την κατάσταση του νευρώνα. Αυτή η απόφαση λαμβάνεται μέσω της σιγμοειδούς συνάρτησης στο στρώμα (layer) που ονομάζεται πύλη της λήθης (forget) και η έξοδος αυτού παράγει έναν αριθμό μεταξύ μηδέν (0) και ένα (1), για κάθε αριθμό της κατάσταση, C_{t-1}. Το ένα (1) δείχνει ότι η μνήμη πρέπει να διατηρηθεί και το μηδέν (0), δείχνει ότι η μνήμη πρέπει να διαγραφεί εντελώς.



2. Αποφασίζει ποιες νέες πληροφορίες θα εγγραφούν στη μνήμη. Αυτή είναι μία διαδικασία δύο βημάτων. Πρώτα, το σιγμοειδές στρώμα που ονομάζεται στρώμα πύλης εισόδου i_t , αποφασίζει τις θέσεις που θα γράψει τις πληροφορίες που χρειάζεται να παραμείνουν στη μνήμη. Στη συνέχεια, ένα στρώμα με συνάρτηση υπερβολικής εφαπτομένης (tanh), δημιουργεί το νέο υποψήφιο C' της κατάστασης του νευρώνα, όπου οι πληροφορίες πρέπει εγγραφούν.



3. Ενημέρωση κατάστασης μνήμης. Η παλιά κατάσταση μνήμης πολλαπλασιάζεται με f_t , διαγράφοντας πράγματα που κρίθηκαν να ξεχαστούν. Στη συνέχεια, η πληροφορία που υπολογίστηκε στο βήμα 2 προστίθεται, για να δημιουργηθεί η νέα κατάσταση μνήμης του νευρώνα, η οποία θα μεταφέρει πληροφορίες στον επόμενο.



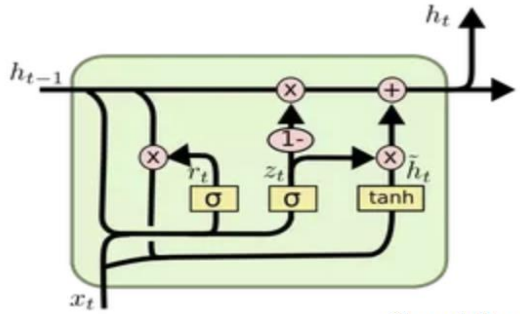
4. Κατάσταση μνήμης εξόδου. Η τελική έξοδος της κατάστασης νευρώνα, εξαρτάται από την τρέχουσα είσοδο και την ενημερωμένη (Ct) μήτρα πληροφοριών του νευρώνα. Πρώτα, χρησιμοποιείται ένα σιγμοειδές στρώμα για να αποφασίσει ποια μέρη της κατάστασης του νευρώνα πρόκειται να εξάγουμε (o_t) Στη συνέχεια, η κατάσταση (h_t) διέρχεται μέσω της συνάρτησης tanh(C_t) που πολλαπλασιάζεται με τη σιγμοειδή πύλη εξόδου(o_t) και περνά ως είσοδος, στο επόμενο χρονικό βήμα του δικτύου μας.

Τα LSTM μπορούν να χρησιμοποιηθούν για πρόβλεψη ακολουθιών καθώς και για ταξινόμηση ακολουθιών. Για παράδειγμα, μπορούμε να προβλέψουμε μελλοντικές τιμές μετοχών. Επίσης, μπορούμε να χρησιμοποιήσουμε LSTM για τη δημιουργία ταξηνομιτών που θα προβλέψουν εάν ένα σήμα εισόδου από κάποιο σύστημα παρακολούθησης της υγείας είναι θανατηφόρο ή μη θανατηφόρο σήμα, σαν ένας δυαδικός ταξηνομιτής. Μπορούμε ακόμη να δημιουργήσουμε έναν ταξινομητή εγγράφων κειμένου με LSTM. Η ακολουθία των λέξεων θα πήγαινε ως είσοδος στο επίπεδο LSTM και η κρυφή κατάσταση του LSTM θα συνδεόταν με ένα πυκνό επίπεδο softmax ως ταξινομητή.

1.4.3 Συσσωρευμένα LSTM

Αν θέλουμε να μάθουμε για την ιεραρχική αναπαράσταση διαδοχικών δεδομένων, μπορεί να χρησιμοποιηθεί μία στοίβα επιπέδων LSTM. Κάθε στρώμα LSTM εξάγει μία ακολουθία διανυσμάτων και όχι έναν απλό φορέα για κάθε στοιχείο της ακολουθίας, ο οποίος θα χρησιμοποιηθεί ως είσοδος σε ένα επόμενο στρώμα LSTM. Αυτή η ιεραρχία κρυφών επιπέδων επιτρέπει μία πιο περίπλοκη αναπαράσταση των διαδοχικών δεδομένων μας. Τα στοιβαγμένα μοντέλα LSTM μπορούν να χρησιμοποιηθούν για μοντελοποίηση πολύπλοκων δεδομένων χρονοσειρών πολλαπλών παραλλαγών.

Μία παραλλαγή του LSTM είναι η Gated Recurrent Unit, ή GRU, που εισήχθη από τους Cho, et al. (2014) . Συνδυάζει τις πύλες λήθης και εισόδου σε μία μόνο «πύλη ενημέρωσης». Συγχωνεύει επίσης την κατάσταση νευρώνα και την κρυφή κατάσταση και κάνει κάποιες άλλες αλλαγές. Το μοντέλο που προκύπτει είναι απλούστερο από τα τυπικά μοντέλα LSTM και έχει γίνει όλο και πιο δημοφιλές.

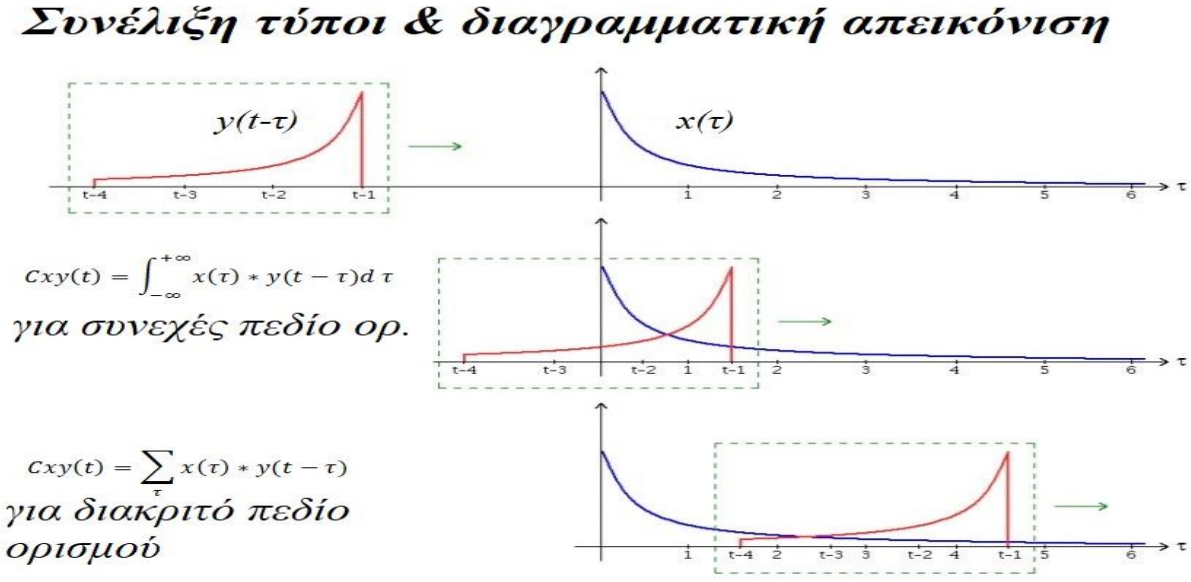


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

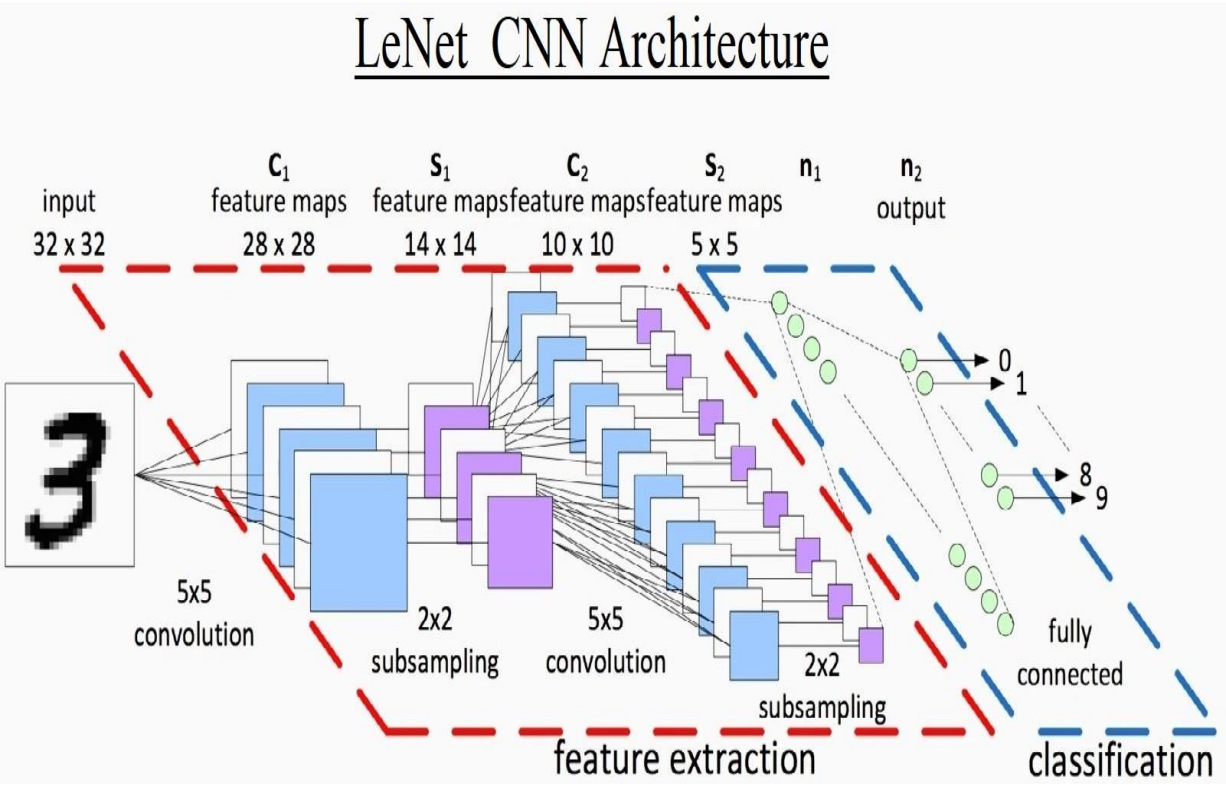
Gated Recurrent Unit

1.5 Συνελικτικά Νευρωνικά Δίκτυα (Convolution Neural Networks – CNN)

Συνελικτικά νευρωνικά δίκτυα, ή CNN, είναι μία οικογένεια μοντέλων που εμπνεύστηκαν από το πώς λειτουργεί ο οπτικός φλοιός του ανθρώπινου εγκεφάλου κατά την αναγνώριση αντικειμένων. Η ανάπτυξη των CNN ανάγεται στη δεκαετία του 1990, όταν ο Yann LeCun και οι συνάδελφοί του πρότειναν μία νέα αρχιτεκτονική νευρωνικού δικτύου για την ταξινόμηση χειρόγραφων ψηφίων από εικόνες. Διακρίνονται ως ένας συγκεκριμένος τύπος νευρωνικού δικτύου που επεξεργάζεται δεδομένα με τοπολογία τύπου πλέγματος. Τα CNN χρησιμοποιούνται ευρέως στον τομέα της όρασης υπολογιστών και ξεπερνούν τις περισσότερες από τις παραδοσιακές τεχνικές υπολογιστικής όρασης που χρησιμοποιούμε. Τα CNN συνδυάζουν τη λειτουργία συνέλιξης και τα νευρικά δίκτυα. Το όνομα συνελικτικό νευρωνικό δίκτυο σημαίνει ότι το δίκτυο χρησιμοποιεί μία μαθηματική λειτουργία που ονομάζεται συνέλιξη και περιλαμβάνει πολλαπλασιασμό πινάκων (elementwise).



Ο κύριος σκοπός της λειτουργίας συνέλιξης είναι η εξαγωγή πληροφοριών ή χαρακτηριστικών από μία εικόνα. Οποιαδήποτε εικόνα θα μπορούσε να θεωρηθεί ως πίνακας τιμών. Μία συγκεκριμένη ομάδα τιμών σε αυτόν τον πίνακα θα σχηματίσει ένα χαρακτηριστικό. Οι αρχικές διεργασίες των CNNs, αφορούν τη μείωση των διαστάσεων μίας εικόνας (πλάτος, ύψος και αριθμός καναλιών). Όσο μεγαλύτερη είναι η εικόνα, τόσο περισσότερος χρόνος απαιτείται.



Σχήμα 24. Αρχιτεκτονική LeNet CNN

Επειδή ο υπολογιστής μας αναγνωρίζει μόνο αριθμητικά στοιχεία, ο απλούστερος τρόπος για την ομαλοποίηση των εισόδων είναι να λάβουμε την τιμή κάθε εικονοστοιχείου και να τη διαιρέσουμε με 255, έτσι ώστε να καταλήξουμε σε τιμές που κυμαίνονται μεταξύ 0 και 1. Διαφορετική μεθοδολογία για την ομαλοποίηση μίας εικόνας, είναι η τεχνική μέσου κεντραρίσματος . Η απόφαση να επιλέξουμε το ένα ή το άλλο είναι τις περισσότερες φορές, θέμα προτίμησης.

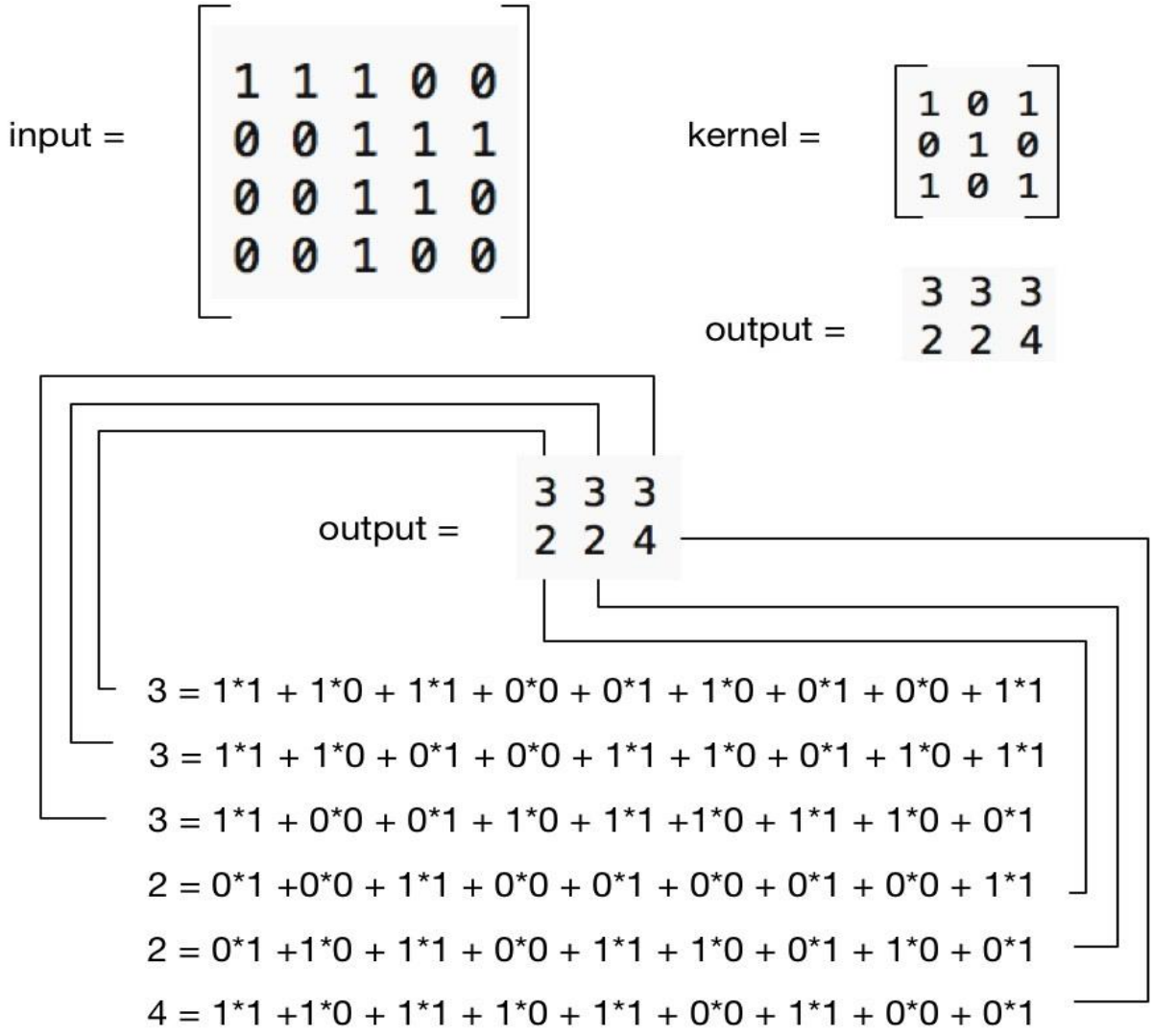
1.5.1 Η λειτουργία της συνέλιξης

Η παρουσίαση γίνεται μέσω του παραδείγματος που ακολουθεί :

Σαρώνουμε την αρχική εικόνα (πίνακας εισόδων 4X5 - inputs) με τη μήτρα φίλτρο ή πυρήνας (μήτρα 3X3 - kernel) και κάθε φορά μετακινούμαστε 1 βήμα δεξιά , έπειτα 1 βήμα κάτω και δεξιά πάλι (stride : 1,1).

Αρχίζουμε με τον 1ο υπο-πίνακα 3X3 των εισόδων (inputs) και για κάθε θέση της εικόνας, εκτελούμε πολλαπλασιασμό (element-wise) με τα στοιχεία της μήτρας kernel.

Σε κάθε βήμα της σάρωσης καταλήγουμε σε έναν αριθμό ως έξοδο. Στο τέλος θα σχηματιστεί ο πίνακας εξόδου (ouput).



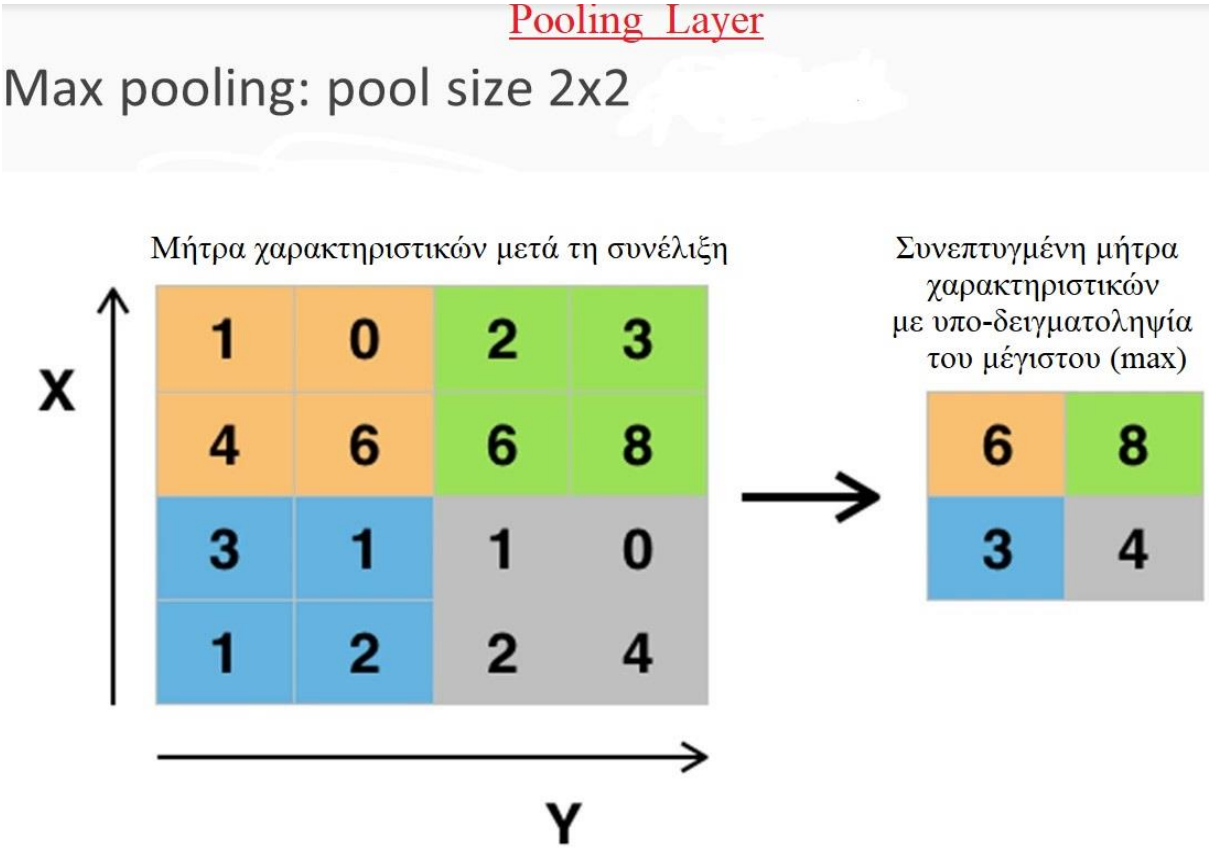
Σχήμα 25. Η λειτουργία της συνέλιξης. Ο πίνακας εξόδου (output) ονομάζεται συνεπτυγμένο χαρακτηριστικό ή χάρτης χαρακτηριστικών.

Στον κάτωθι πίνακα η συνέλιξη του φίλτρου και των εισόδων του πίνακα στα αριστερά, παράγει ως έξοδο πίνακα μικρότερων διαστάσεων, ο οποίος θα αποτελέσει τη μήτρα δεδομένων εισόδου στο επόμενο επίπεδο, όπου μία νέα διαδικασία συνέλιξης θα παράγει ως έξοδο μικρότερων διαστάσεων πίνακα χαρακτηριστικών.

Σχήμα 26. Πρώτο συνελικτικό στρώμα, για την αναγνώριση του αριθμού "7".

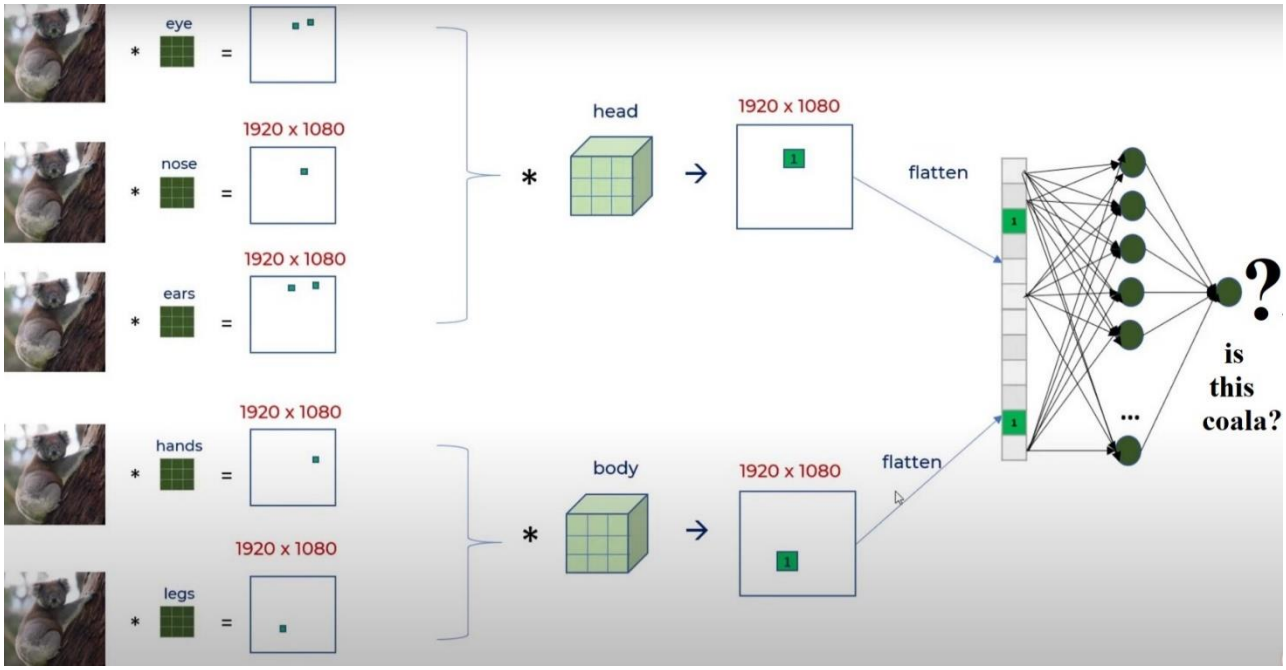
1.5.2 Ομαδοποίηση (pooling - subsampling)

Τα στρώματα ομαδοποίησης ή υπο-δειγματοληψίας (pooling - subsampling) στα CNNs, είναι ένας πολύ ταχύτερος μηχανισμός για μείωση του μεγέθους του πίνακα εισόδων. Βοηθούν στην αντιμετώπιση του φαινομένου της υπερβολικής προσαρμογής (overfitting) και στη βελτίωση της απόδοσης, μειώνοντας των αριθμό των εισόδων, αλλά διατηρώντας σημαντικές πληροφορίες. Οι κύριες επιλογές γίνονται με βάση τη μέση (average pooling) ή τη μέγιστη τιμή (max pooling) .



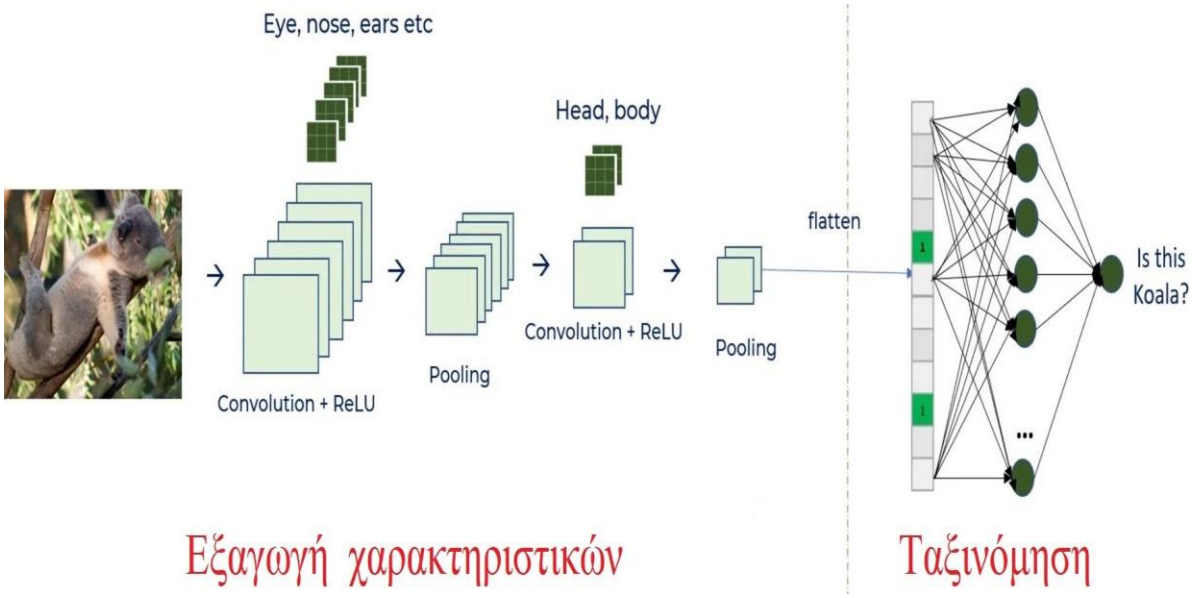
Σχήμα 27. Max pooling με υπο-δειγματοληψία του μέγιστου

Τα φίλτρα αποτελούν τα βάρη (weights) του δικτύου τα οποία αναπροσαρμόζονται και ποικίλουν ανάλογα το είδος της εικόνας που αναλύουμε. Υπάρχουν φίλτρα που διακρίνουν τις οριζόντιες ή τις κάθετες γραμμές, τους κύκλους, τις γωνίες, αλλά και πιο σύνθετα, όπως για τη διάκριση της μύτης, των αυτιών ακόμα και του προσώπου. Καθώς περνάμε από αυτήν τη διαδικασία, το δίκτυό μας θα αναπτύξει έναν δισδιάστατο χάρτη ενεργοποίησης για να παρακολουθεί την απόκριση του συγκεκριμένου φίλτρου σε μία δεδομένη θέση. Το δίκτυο θα μάθει να διατηρεί φίλτρα που ενεργοποιούνται όταν φτάνουν στην άκρη ενός σχήματος, ή ίσως στο χρώμα ενός αντικειμένου. Κάθε φίλτρο για ένα συγκεκριμένο συνελικτικό επίπεδο μας αφήνει με ατομικούς χάρτες ενεργοποίησης. Αν έχουμε έξι φίλτρα για μία εικόνα, θα έχουμε έξι χάρτες ενεργοποίησης, καθένας από τους οποίους εστιάζει σε κάτι διαφορετικό μέσα στην εικόνα. Στα πρώιμα κρυφά επίπεδα (early hidden layers) ενός CNN, εξάγουμε χαμηλού επιπέδου χαρακτηριστικά, όπως άκρες, καμπύλες κα.



Σχήμα 28. Εξαγωγή επιπλέον χαρακτηριστικών

Στα ενδότερα του δικτύου, η επεξεργασία των εξόδων μας δίνει υψηλότερου επιπέδου χαρακτηριστικά, όπως μύτη, χέρια, μάτια, κεφάλι, σώμα και άλλα παρόμοια γενικά χαρακτηριστικά. Οι φυσικές εικόνες έχουν μία ιδιότητα όπου η στατιστική που διέπει ένα patch (παράθυρο) της εικόνας, είναι η ίδια σε κάθε άλλο μέρος της. Άρα τα χαρακτηριστικά που μαθαίνουμε για κάποιο patch, μπορούμε να τα εφαρμόσουμε και σε άλλα μέρη της εικόνας.



Σχήμα 29. Εξαγωγή και ταξινόμηση

Οι γραμμικές μέθοδοι στις περισσότερες περιπτώσεις CNNs, αδυνατούν να δώσουν ικανοποιητικά αποτελέσματα. Το πλεονέκτημα των μη γραμμικών νευρωνικών δικτύων, εκδηλώνεται με τη συμμετοχή των γνωστών μη γραμμικών συναρτήσεων tanh, sigmoid και Rectified Linear Unit (ReLU). Πιο κατάλληλη θεωρείται η ReLU, με την οποία οι αρνητικές τιμές μηδενίζονται. Η μετατροπή αυτή βοηθά στην απλοποίηση και επιτάχυνση του αλγορίθμου backpropagation, μειώνοντας τις πράξεις που απαιτούνται.

Την φάση της εξαγωγής των χαρακτηριστικών, διαδέχεται η φάση της ταξινόμησης. Τα εξαγόμενα διανύσματα χαρακτηριστικών συνδέονται πλήρως σε ένα κλασικό νευρωνικό δίκτυο. Εκτελείται η διεργασία της ταξινόμησης σε κλάσεις, υπολογίζοντας τις πιθανότητες που έχει η εικόνα της εισόδου να ανήκει σε κάποια κλάση. Η τελική έξοδος προκύπτει ως η κλάση, με τη μεγαλύτερη πιθανότητα.

1.5.3 Εκπαίδευση CNNs

Η εκπαίδευση των CNNs αφορά την κατηγορία μάθησης με επίβλεψη και γίνεται με τον αλγόριθμο backpropagation με τη βοήθεια της μεθόδου κατάβασης κλίσης, για την εύρεση των βέλτιστων τιμών των πινάκων χαρακτηριστικών. Αρχικά ορίζουμε τα χαρακτηριστικά, τον αριθμό των επαναλήψεων εκπαίδευσης, τον αριθμό των πακέτων εικόνων που θα επεξεργαζόμαστε, μία παράμετρο “β”, που αντιπροσωπεύει τον ρυθμό εκμάθησης και έναν δείκτη του βαθμού στον οποίο εκπαιδεύεται το δίκτυο σε κάθε επανάληψη.

Η διαδικασία της οπισθοδιάδοσης σφάλματος συνίσταται στη διάδοση κλίσεων στα προηγούμενα επίπεδα του δικτύου για τη αναπροσαρμογή των βαρών, των φίλτρων δηλαδή που έχουν υπολογιστεί. Αν τα βάρη αλλάζουν ελάχιστα την τιμή τους λόγω της μειωμένης διάδοσης της κλίσης (εκπαιδεύονται ανεπαρκώς), η λύση μπορεί να δοθεί από τη διάθεση ενός μεγάλου αριθμού δειγμάτων σχετικών εικόνων (5000 ανά κλάση) από μεγάλη βάση δεδομένων και μετά από κάποιο χρόνο εκπαίδευσης, το δίκτυο καταφέρνει να αποκτήσει μία αποδοτική λειτουργικότητα. Στη συνέχεια μπορούμε να χρησιμοποιήσουμε ως εισόδους από τα δεδομένα που έχουμε στη διάθεσή μας.

1.5.3.1 Προ εκπαίδευση

Όταν τα δεδομένα εκπαίδευσης είναι ανεπαρκή, απαιτείται προ-εκπαίδευση και η τεχνική που ακολουθείται είναι η παροχή στο Δίκτυο διανυσμάτων-εικόνων μεγάλης βάσης δεδομένων από τομέα σχετικό με την εφαρμογή που θέλουμε να δημιουργήσουμε, ώστε να επέλθει σύγκλιση των παραμέτρων του Δικτύου. Στη συνέχεια, κατά την εκπαίδευση, δίνονται ως είσοδοι εικόνες-διανύσματα από τα δεδομένα που έχουμε στη διάθεσή μας. Στην περίπτωση των εικόνων, ένα επιπλέον βήμα που μπορεί να ακολουθηθεί, είναι η επέκταση της βάσης δεδομένων μέσα από επεξεργασία των εικόνων εκμεταλλευόμενοι του στοιχείου της "τοπικότητας" των Συνελικτικών Νευρωνικών Δικτύων. Έτσι, για κάθε εικόνα μπορούμε να δημιουργήσουμε μερικά ελαφρώς παραλλαγμένα αντίγραφα, όπου έχουμε αλλάξει τις τιμές αριθμού τυχαίων pixel πάνω στην εικόνα, πχ ανάθεση σε ορισμένα pixel της τιμής "0" ή "1", περιστροφές των εικόνων για ελάχιστες μοίρες (-10 εως +10 μοίρες) και μετατόπιση των εικόνων σε δυο διευθύνσεις με βήμα ένα μικρό αριθμό pixels (translation). Με αυτό τον τρόπο, δίνουμε στο Δίκτυο μία άλλη "οπτική" της ίδιας πληροφορίας, όπως συμβαίνει και στις εικόνες RGB όπου το Δίκτυο έχει στη διάθεσή του τρείς διαφορετικές εκδοχές της ίδιας εικόνας, με αποτέλεσμα η εκπαίδευσή του να γίνεται πιο αποτελεσματική.

Γενικά η προ-εκπαίδευση:

 Κάνει τη βελτιστοποίηση του Δικτύου πιο εύκολη

 Μειώνει την υπερ-προσαρμογή (overfitting)

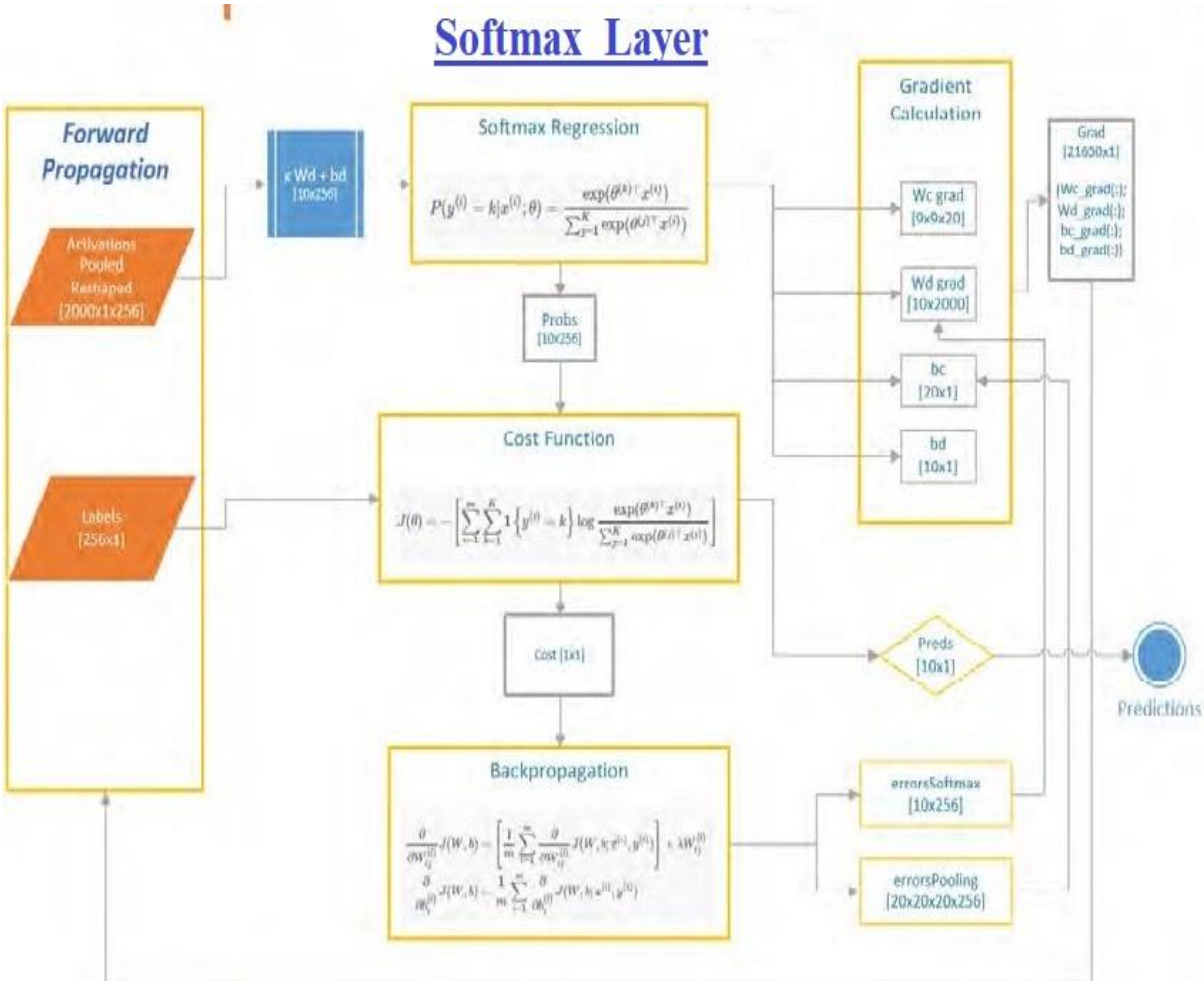
Σε άλλες μορφές νευρωνικών δικτύων, για παράδειγμα στους Autoencoders και τις RBMs, η προ-εκπαίδευση είναι αρκετά σημαντική αφού ένα από τα θεμελιώδη ζητήματα για την ορθή εκπαίδευσή τους αποτελεί η ανάθεση κατάλληλων αρχικών τιμών των βαρών τους.

1.5.3.2 Εκπαίδευση

Προκειμένου να εκπαιδεύσουμε το νευρωνικό δίκτυο, ορίζουμε κάποιες επιλογές όπως για παράδειγμα τον αριθμό των επαναλήψεων (epoch) που χρειαζόμαστε για την εκπαίδευση, Το κομμάτι του forward propagation περιλαμβάνει ένα συνελικτικό επίπεδο με 20 νευρώνες συνέλιξης, εισαγωγή μη γραμμικότητας τύπου ReLU και ένα επίπεδο pooling. Κάθε ένα από αυτά δέχεται είτε είσοδο-εικόνα, είτε είσοδο-χαρακτηριστικό διάνυσμα με σκοπό την εκτέλεση συνελίξεων ώστε να παραχθούν τα χαρακτηριστικά διανύσματα.

1.5.4 Softmax layer

Τα περιεχόμενα του Softmax Layer (σχήμα 30) συνίστανται από μία συνάρτηση Λογιστικής Παλινδρόμησης, μία Cost function και έναν υπολογιστή της κλίσης για Back Propagation. Τα δεδομένα που δέχεται η Λογιστική Συνάρτηση είναι διαστάσεων 10x256, δηλαδή 256 τιμές για κάθε μία από τις 10 κλάσεις. Σε αυτό το σημείο υπολογίζονται οι πιθανότητες (probs) κάθε μία εικόνα να ανήκει σε μία συγκεκριμένη κλάση και ακολουθεί η Cost Function, δηλαδή μία συνάρτηση κόστους (cross entropy objective), συνάρτηση η οποία δέχεται ως είσοδο τις επισημάνσεις και τις πιθανότητες που έχουν υπολογιστεί για κάθε εικόνα-διάνυσμα και υπολογίζει ένα αριθμό-δείκτη της ορθής εκτίμησης του περιεχομένου κάθε εικόνας. Το τελικό αποτέλεσμα είναι ένας πίνακας 10x1 για κάθε εικόνα εισόδου (test), ο οποίος μας δείχνει πόση είναι η πιθανότητα να ανήκει το ψηφίο της εικόνας εισόδου σε κάθε κλάση.



Σχήμα 30. Softmax Layer

Σημείωση

Οι εφαρμογές της τεχνητής νοημοσύνης (AI), υποκαθιστούν με επιτυχία και σχετικά γρήγορα, ολοένα και περισσότερες απλές ανθρώπινες λειτουργίες. Η όραση, η κίνηση, η ταξινόμηση, η εκτίμηση, η πρόβλεψη καταστάσεων από μία μηχανή, απαιτεί πολύπλοκες υπολογιστικές διεργασίες και τεράστιο αριθμό δεδομένων για τον έλεγχο της προσαρμογής σε πραγματικές συνθήκες. Οι τεχνικοί της Google με την παρουσίαση του Tensorflow API, έχουν καταφέρει να συμπτύξουν σε μεγάλο βαθμό, την επεξεργασία που απαιτούν τα δίκτυα τεχνητής νοημοσύνης για να είναι αποτελεσματικά. Με σχετικά μικρό αριθμό εντολών στοTensorflow API, κατασκευάζονται, εκπαιδεύονται και ελέγχονται όλα τα δίκτυα που αναλύσαμε σε αυτό το κεφάλαιο.

ΚΕΦΑΛΑΙΟ 2ο

TensorFlow

Η βιβλιοθήκη TensorFlow ξεκίνησε το 2011, με την ονομασία DisBelief, αρχικά ως ένα εσωτερικό έργο κλειστού κώδικα της Google. Το DisBelief ήταν ένα σύστημα μηχανικής μάθησης που χρησιμοποίησε νευρωνικά δίκτυα βαθιάς μάθησης. Αυτό το σύστημα μετατράπηκε σε TensorFlow, το οποίο κυκλοφόρησε στην κοινότητα των προγραμματιστών με άδεια χρήσης ανοιχτού κώδικα Apache 2.0, στις 9 Νοεμβρίου 2015. Μετά την εμφάνιση της έκδοσης 1.0.0 τον Φεβρουάριο του 2017, η Google κυκλοφόρησε τον Μάρτιο του 2019 την έκδοση TensorFlow 2.0.0, συνεχίζοντας μέχρι σήμερα την ανάπτυξη και τη βελτίωση του TensorFlow, καθιστώντας το πιο φιλικό και προσβάσιμο προς το χρήστη. Παρέχοντας έτσι τη δυνατότητα χειρισμού διαφορετικών υπολογιστικών αρχιτεκτονικών (CPU, GPU και TPUs). Μπορούμε να διαβάσουμε το πρωτότυπο έγγραφο για το έργο εδώ: http://download.tensorflow.org/paper/whitepaper2015.pdf

Η βιβλιοθήκη παρέχει έναν μοναδικό τρόπο διαχείρισης προβλημάτων, επιτρέποντάς μας να επιλύσουμε τα ζητήματα μηχανικής μάθησης (ML) αρκετά αποτελεσματικά. Σήμερα η μηχανική μάθηση χρησιμοποιείται σχεδόν στους περισσότερους τομείς της ζωής και της εργασίας, με διαδεδομένες εφαρμογές που αφορούν την "όραση" του υπολογιστή, την αναγνώριση ομιλίας, τις μεταφράσεις γλώσσας και άλλα. Γίνεται ευρεία χρήση στην αναγνώριση προσώπου και δακτυλικών αποτυπωμάτων ή την ταυτοποίηση οχήματος μέσω πινακίδας κυκλοφορίας, για την ανάκτηση πληροφοριών από μηχανές αναζήτησης όπως αναζήτηση κειμένου για την εύρεση εικόνων. Έχει συμβάλλει αποτελεσματικά στην ιατρική διάγνωση καρκινικών όγκων ή χρόνιας νόσου, καθώς και στην επεξεργασία φυσικής γλώσσας για εφαρμογές όπως ετικέτες φωτογραφιών.

2.1 Λειτουργία TensorFlow

Η καλύτερη υποστήριξη προγραμματισμού για το TensorFlow παρέχεται από την Python 3.6 - 3.9 ενώ υπάρχουν επιπλέον βιβλιοθήκες για Java, C και Go και για άλλες γλώσσες που βρίσκονται υπό ενεργή ανάπτυξη. Ενώ το TensorFlow μπορεί να εκτελεστεί στην CPU, οι περισσότεροι αλγόριθμοι τρέχουν γρηγορότερα αν υποβληθούν σε επεξεργασία σε GPU και υποστηρίζεται με κάρτες γραφικών Nvidia Compute Capability 3.5 ή υψηλότερη (προτιμάται

όταν τρέχουμε πολύπλοκα δίκτυα). Υπάρχουν δύο ξεχωριστές εκδόσεις του TensorFlow, για CPU και για GPU στη διεύθυνση : https://www.tensorflow.org/install

Αυτό το τελευταίο απαιτεί την εγκατάσταση των αριθμητικών βιβλιοθηκών CUDA και CuDNN με την αντίστοιχη έκδοση Python. Προς το παρόν, το TensorFlow δοκιμάζεται και υποστηρίζεται στα ακόλουθα συστήματα 64-bit: Ubuntu 16.04 ή μεταγενέστερη έκδοση, macOS 10.12.6 (Sierra) ή μεταγενέστερη έκδοση (ωστόσο, δεν υπάρχει υποστήριξη GPU), Raspbian 9.0 ή μεταγενέστερη έκδοση και Windows 7 ή μεταγενέστερη έκδοση.

Ο κύριος τομέας της έρευνας και ανάπτυξης του TemsorFlow είναι στις εφαρμογές Deep Neural Networks (DNN), όπου έχει χρησιμοποιηθεί σε διάφορους τομείς όπως αναγνώριση φωνής και ήχου. Για παράδειγμα, χρησιμοποιείται στους πλέον διαδεδομένους βοηθούς που ενεργοποιούνται με τη φωνή, επίσης σε εφαρμογές που βασίζονται σε κείμενο, όπως σε έξυπνους μεταφραστές γλωσσών, είναι ευρέως διαδεδομένο στην αναγνώριση εικόνας, για παράδειγμα στην ανίχνευση και τη διάγνωση καρκίνου, καθώς και σε εφαρμογές χρονοσειρών.

Το **TensorFlow.js** είναι μία συλλογή από API που μας επιτρέπουν να δημιουργήσουμε και να εκπαιδεύσουμε μοντέλα χρησιμοποιώντας σε χαμηλό επίπεδο τη βιβλιοθήκη γραμμικής άλγεβρας JavaScript. Τα μοντέλα δηλαδή, μπορούν να εκπαιδευτούν και να εκτελεστούν σε ένα πρόγραμμα περιήγησης (browser).

Το **TensorFlow Lite** είναι μία ελαφριά έκδοση του TensorFlow για κινητές και ενσωματωμένες συσκευές. Αποτελείται από έναν διερμηνέα χρόνου εκτέλεσης και ένα σύνολο βοηθητικών προγραμμάτων. Η ιδέα είναι να εκπαιδεύσουμε ένα μοντέλο σε μηχανή υψηλότερης ισχύος και στη συνέχεια να μετατρέψουμε το μοντέλο μας σε " .tflite " μορφή, χρησιμοποιώντας τα βοηθητικά προγράμματα. Στη συνέχεια, φορτώνουμε το μοντέλο στη συσκευή της επιλογής μας. Το TensorFlow Lite υποστηρίζεται σε Android και iOS με C ++ API και διαθέτει Java wrapper για Android. Εάν μία συσκευή Android υποστηρίζει το API Neural Networks (ANN) Android για επιτάχυνση υλικού, τότε ο διερμηνέας θα το χρησιμοποιήσει, διαφορετικά θα προεπιλεγεί στην CPU για εκτέλεση.

Το **TensorFlow Hub** είναι μία βιβλιοθήκη που έχει σχεδιαστεί για να προωθεί τη δημοσίευση, την ανακάλυψη και τη χρήση επαναχρησιμοποιήσιμων ενοτήτων, μοντέλων μηχανικής μάθησης. Σε αυτό το πλαίσιο, μία ενότητα είναι ένα αυτόνομο κομμάτι ενός γραφήματος TensorFlow μαζί με τα βάρη και άλλα στοιχεία. Η ενότητα μπορεί να επαναχρησιμοποιηθεί σε διαφορετικές εργασίες με μία μέθοδο γνωστή ως μεταφορά μάθησης. Η ιδέα είναι να εκπαιδεύσουμε ένα μοντέλο σε ένα μεγάλο σύνολο δεδομένων και στη συνέχεια να επαναπροσδιορίσουμε την κατάλληλη ενότητα για τη διαφορετική αλλά σχετική εργασία μας. Έτσι μπορούμε να εκπαιδεύσουμε ένα μοντέλο με ένα μικρότερο σύνολο δεδομένων, να βελτιώσουμε τη γενίκευση και μπορούμε να επιταχύνουμε σημαντικά την εκπαίδευση.

Το **TensorFlow Extended (TFX)** είναι μία πλατφόρμα μηχανικής μάθησης γενικής χρήσης με βάση το TensorFlow. Οι βιβλιοθήκες που έχουν κυκλοφορήσει σε ανοιχτό κώδικα μέχρι σήμερα περιλαμβάνουν το TensorFlow Transform, το TensorFlow Model Analysis και το TensorFlow Serving.

Το **Keras(tf.keras)** είναι ένα υψηλού επιπέδου API νευρωνικών δικτύων, γραμμένο σε Python, το οποίο διασυνδέεται με το TensorFlow. Το tf.keras είναι φιλικό προς το χρήστη, αρθρωτό και επεκτάσιμο. Υποστηρίζει τόσο συμβατικά όσο και επαναλαμβανόμενα δίκτυα και λειτουργεί σε CPU και GPU.

Το **TensorBoard** είναι μία σειρά εργαλείων οπτικοποίησης, που υποστηρίζουν την κατανόηση, τον εντοπισμό σφαλμάτων και τη βελτιστοποίηση των προγραμμάτων TensorFlow. Μπορούμε να χρησιμοποιήσουμε το TensorBoard για να απεικονίσουμε διάφορες μετρήσεις του μοντέλου μας κατά τη διάρκεια της εκπαίδευσης.

2.2 Google Colaboratoryoratory

Ο πιο βολικός τρόπος χρήσης του TensorFlow, είναι το Google Colaboratory. Είναι ένα δωρεάν διαδικτυακό περιβάλλον που βασίζεται στο Jupyter notebook και δεν απαιτεί ρύθμιση, καθώς συνοδεύεται από όλες τις εξαρτήσεις που είναι προ-εγκαταστημένες. Το Google Colaboratory παρέχει έναν εύκολο και βολικό τρόπο στους χρήστες να γράψουν κώδικα

TensorFlow στο πρόγραμμα περιήγησής τους, χωρίς να χρειάζεται να ανησυχείτε για κάθε είδους εγκαταστάσεις. Μπορούμε να χρησιμοποιήσουμε το Google Colaboratoru στη διεύθυνση https://colaboratory.research.google.com/notebooks/intro.ipynb#recent=true

Μας δίνεται η επιλογή από την κονσόλα, η οποία περιέχει τις ακόλουθες πέντε καρτέλες:

1. **Παραδείγματα**: Εμφανίζει τους προεπιλεγμένους φορητούς υπολογιστές που παρέχονται στο Colaboratory.

2. **Πρόσφατα**: Τα τελευταία σημειωματάρια στα οποία δούλεψε ο χρήστης.

3. **Google Drive**: Τα σημειωματάρια που συνδέονται με το λογαριασμό Google Drive του χρήστη

4. **GitHub**: Η επιλογή σύνδεσης των φορητών υπολογιστών που υπάρχουν στον λογαριασμό GitHub του χρήστη.

5. **Μεταφόρτωση**: Η επιλογή μεταφόρτωσης ενός νέου ipynb ή ενός αρχείου GitHub.

Άλλη επιλογή είναι να κάνετε κλικ στο «Αρχείο ~ Νέο Σημειωματάριο» και θα εμφανιστεί ένα νέο σημειωματάριο. Το Colaboratory έρχεται προεγκατεστημένο με το TensorFlow. Ένα άλλο μεγάλο πλεονέκτημα της χρήσης του Colaboratory είναι ότι μας επιτρέπει να δημιουργήσουμε τα μοντέλα μας σε GPU, χρησιμοποιώντας Keras, TensorFlow και PyTorch. Παρέχει επίσης 12 GB RAM, με χρήση έως και 12 ώρες.

2.3 Τα βασικά του TensorFlow

2.3.1 Τανυστές - Tensors

Το TensorFlow παίρνει το όνομά του από τους τανυστές (tensors). Ένας τανυστής είναι μία γενίκευση διανυσμάτων και πινάκων σε υψηλότερες διαστάσεις. Ο βαθμός ενός τανυστή είναι ο αριθμός των δεικτών που απαιτούνται για τον μοναδικό προσδιορισμό κάθε στοιχείου αυτού του τανυστή. Ένα βαθμωτό (ένας απλός αριθμός), είναι ένας τανυστής της τάξης 0, ένα

διάνυσμα είναι ένας τανυστής της τάξης 1, ένας πίνακας είναι ένας τανυστής της τάξης 2 και ένας τρισδιάστατος πίνακας είναι ένας τανυστής της τάξης 3. Ένας τανυστής έχει έναν τύπο δεδομένων και ένα σχήμα (όλα τα στοιχεία δεδομένων σε έναν τανυστή πρέπει να έχουν τον ίδιο τύπο). Το παράδειγμα ενός 4-διάστατου τανυστή (rank 4) είναι μία εικόνα όπου οι 4 διαστάσεις της, έχουν τα χαρακτηριστικά batch, height, width, και color (για παράδειγμα):

```
import tensorflow as tf

image= tf.zeros([2, 3, 2, 3]) #παραδειγμα 4ου βαθμού tensor, με χαρακτηριστικά batch, hight, weight, color

print(image)
```

```
tf.Tensor(
      1,  2,  3                          1,  2,  3
[ [ [ [0. 0. 0.]     '1'        [ [ [ [0. 0. 0.]     '1'
      [0. 0. 0.] ]   '2'            [0. 0. 0.] ]   '2'
    [ [0. 0. 0.]     '1'          [ [0. 0. 0.]     '1'
      [0. 0. 0.] ]   '2'            [0. 0. 0.] ]   '2'
    [ [0. 0. 0.]     '1'          [ [0. 0. 0.]     '1'
      [0. 0. 0.] ] ] '2'            [0. 0. 0.] ] ] '2'
 Shape= (2, 3, 2, 3), dtype=float32)
```

2.3.2 Γενική ροή των αλγορίθμων TensorFlow.

2.3.2.1 Εισαγωγή συνόλων δεδομένων (inputs)

Όλοι οι αλγόριθμοι μηχανικής μάθησης έχουν άμεση εξάρτηση από τα σύνολα δεδομένων, που χρησιμοποιούνται στις εισόδους (inputs). Τα δεδομένα μπορούμε να τα δημιουργήσουμε ή να χρησιμοποιήσουμε μία εξωτερική πηγή, από σύνολα δεδομένων. Πολλές φορές, μας δίνεται η δυνατότητα να έχουμε πρόσβαση σε δημόσια σύνολα δεδομένων, ανάλογα τον τομέα που επιχειρούμε. Μερικές φορές, είναι καλύτερα να βασιζόμαστε στα δεδομένα δημιουργούμε, επειδή μπορούμε να ελέγξουμε τον τρόπο μεταβολής και να επαληθεύσουμε το αναμενόμενο αποτέλεσμα.

```
import tensorflow as tf

import tensorflow_datasets as tfds

import numpy as np

data = tfds.load("iris", split="train")
```

2.3.2.2 Μετασχηματισμός δεδομένων

Γενικά, τα σύνολα δεδομένων εισόδου δεν έχουν την ακριβή μορφή που θέλουμε για αυτό που σκοπεύουμε να επιτύχουμε. Η TensorFlow αναμένει από εμάς να μετατρέψουμε τα δεδομένα σε αποδεκτό σχήμα και τύπο δεδομένων. Στην πραγματικότητα τα δεδομένα συνήθως δεν έχουν τη σωστή διάσταση ή τον τύπο που περιμένουν οι αλγόριθμοί μας και θα πρέπει να τα μετατρέψουμε σωστά προτού τα χρησιμοποιήσουμε. Οι περισσότεροι αλγόριθμοι αναμένουν επίσης κανονικοποιημένα δεδομένα (που συνεπάγονται μεταβλητές των οποίων ο μέσος όρος είναι μηδέν και των οποίων η τυπική απόκλιση είναι ένα). Το TensorFlow προσφέρει ενσωματωμένες λειτουργίες που μπορούν να φορτώσουν τα δεδομένα μας, να τα χωρίσουν σε ομάδες και να μας επιτρέψουν να μεταμορφώσουμε τις μεταβλητές και να μετασχηματήσουμε κάθε ομάδα (batch), χρησιμοποιώντας απλές λειτουργίες NumPy.

```
for batch in data.batch(batch_size, drop_remainder=True):
    labels = tf.one_hot(batch['label'], 3)
    X = batch['features']
    X = (X - np.mean(X)) / np.std(X)
```

2.3.2.3 Διαχωρισμός συνόλου δεδομένων σε σύνολα εκπαίδευσης, δοκιμών και επικύρωσης

Γενικά θέλουμε να δοκιμάσουμε τους αλγόριθμους μας σε διαφορετικά σύνολα από εκείνα στα οποία έχουν εκπαιδευτεί. Οι περισσότεροι αλγόριθμοι απαιτούν επίσης συντονισμό

υπερπαραμέτρων, οπότε αφήνουμε ένα σετ επικύρωσης, για τον προσδιορισμό του καλύτερου συνόλου υπερπαραμέτρων.

2.3.2.4 Ορισμός παραμέτρων αλγορίθμου (υπερπαράμετροι)

Οι αλγόριθμοί μας συνήθως έχουν ένα σύνολο παραμέτρων που πρέπει να διατηρούμε σταθερές, καθ 'όλη τη διάρκεια της διαδικασίας. Για παράδειγμα ο αριθμός των επαναλήψεων, το ποσοστό εκμάθησης ή άλλες σταθερές παράμετροι της επιλογής μας. Θεωρείται καλή πρακτική να αρχικοποιούμε τις παραμέτρους μαζί, χρησιμοποιώντας καθολικές μεταβλητές, έτσι ώστε ο αναγνώστης ή ο χρήστης, να τις βρίσκει εύκολα ως εξής:

```
epochs = 1000

batch_size = 32

input_size = 4

output_size = 3

learning_rate = 0.001
```

2.3.2.5 Αρχικοποίηση μεταβλητών

Το TensorFlow είναι σημαντικό να γνωρίζει τι μπορεί και τι δεν μπορεί να τροποποιήσει. Θα προσαρμόσει όλες τις μεταβλητές (βάρη μοντέλου και σταθερού όρου- bias) κατά τη βελτιστοποίηση, ώστε να ελαχιστοποιήσει μία συνάρτηση απώλειας. Για να επιτευχθεί αυτό, τροφοδοτούμε δεδομένα μέσω μεταβλητών εισαγωγής. Πρέπει να αρχικοποιήσουμε τόσο τις μεταβλητές όσο και τα σύμβολα κράτησης θέσης με μέγεθος και τύπο, έτσι ώστε το TensorFlow να γνωρίζει τι να περιμένει.

```
weights = tf.Variable(tf.random.normal(shape=(input_size,
output_size),   dtype=tf.float32))

biases = tf.Variable(tf.random.normal(shape=(output_size,),
dtype=tf.float32))
```

2.3.2.6 Καθορισμός της δομής του μοντέλου

Αφού έχουμε τα δεδομένα και αρχικοποιήσουμε τις μεταβλητές μας, πρέπει να ορίσουμε το μοντέλο μας. Αυτό γίνεται με τη δημιουργία ενός υπολογιστικού γραφήματος. Το μοντέλο για αυτό το παράδειγμα είναι ένα μοντέλο λογιστικής παλινδρόμησης (b-weights, a-bias)

logit E (Y) = b X + a

```
logits = tf.add(tf.matmul(X, weights), biases)
```

2.3.2.7 Συνάρτηση απώλειας

Αφού ορίσουμε το μοντέλο, πρέπει να είμαστε σε θέση να αξιολογήσουμε την έξοδο. Η συνάρτηση απώλειας είναι πολύ σημαντική καθώς μας λέει πόσο μακριά είναι οι προβλέψεις μας από τις πραγματικές τιμές. Ως παράδειγμα, υλοποιούμε τη συνάρτηση κόστους με logits, η οποία υπολογίζει τη διασταυρούμενη εντροπία softmax μεταξύ logits και ετικετών:

```
loss = tf.reduce_mean( tf.nn.softmax_cross_entropy_with_logits(labels, logits))
```

2.3.2.8 Αρχικοποίηση και εκπαίδευση του μοντέλου

Τώρα που έχουμε τα πάντα στη θέση τους, πρέπει να δημιουργήσουμε ένα παράδειγμα του γραφήματος μας, να τροφοδοτήσουμε τα δεδομένα και να αφήσουμε το TensorFlow να αλλάξει τις μεταβλητές για να προβλέψει καλύτερα τα δεδομένα εκπαίδευσης. Ακολουθεί ένας τρόπος αρχικοποίησης του υπολογιστικού γραφήματος και μέσω πολλαπλών επαναλήψεων, σύγκλισης των βαρών στη δομή του μοντέλου χρησιμοποιώντας τον βελτιστοποιητή στοχαστικής κλίσης (Slope Gradient Descent) με ρυθμό εκμάθησης :

```
optimizer = tf.optimizers.SGD(learning_rate)

with tf.GradientTape() as tape:
```

```
        logits = tf.add(tf.matmul(X, weights), biases)
        loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels, logits))
    gradients = tape.gradient(loss, [weights, biases])
    optimizer.apply_gradients(zip(gradients, [weights, biases]))
```

2.3.2.9 Αξιολόγηση του μοντέλου

Αμέσως μετά τη δημιουργία και την εκπαίδευση του μοντέλου μας, θα πρέπει να το αξιολογήσουμε, εξετάζοντας πόσο καλά προσαρμόζεται με τα νέα δεδομένα, μέσω ορισμένων καθορισμένων κριτηρίων. Αξιολογούμε στο σετ εκπαίδευσης και δοκιμών και αυτές οι αξιολογήσεις θα μας επιτρέψουν να δούμε αν το μοντέλο έχει επιτύχει σε ένα εύρος, από κακή προσαρμογή έως υπερπροσαρμογή (overfiting). Στο παράδειγμα αξιολογούμε την τελική απώλεια και συγκρίνουμε τις προσαρμοσμένες τιμές με τις πραγματικές:

```
print(f"final loss is: {loss.numpy():.3f}")
preds = tf.math.argmax(tf.add(tf.matmul(X, weights), biases), axis=1)
ground_truth = tf.math.argmax(labels, axis=1)
for y_true, y_pred in zip(ground_truth.numpy(), preds.numpy()):
    print(f"real label: {y_true} fitted: {y_pred}")
```

2.3.2.10 Ρύθμιση υπερπαραμέτρων

Τις περισσότερες φορές, θα θέλουμε να επιστρέψουμε και να αλλάξουμε μερικές από τις υπερπαραμέτρους, ελέγχοντας την απόδοση του μοντέλου με βάση τις δοκιμές μας. Στη συνέχεια επαναλαμβάνουμε τα προηγούμενα βήματα με διαφορετικές υπερπαραμέτρους και αξιολογούμε το μοντέλο στο σύνολο επικύρωσης. Στο TensorFlow, πρέπει πρώτα να ρυθμίσουμε τα δεδομένα, τις μεταβλητές εισόδου και τη δομή του μοντέλου, πριν μπορέσουμε να πούμε στο πρόγραμμα να εκπαιδεύσει και να ρυθμίσει τα βάρη του για να βελτιώσει τις προβλέψεις. Το TensorFlow το επιτυγχάνει μέσω υπολογιστικών γραφημάτων. Αυτά τα

υπολογιστικά γραφήματα είναι κατευθυνόμενα γραφήματα χωρίς αναδρομή. Για να γίνει αυτό πρέπει να δημιουργήσουμε μία συνάρτηση απώλειας, για να την ελαχιστοποιήσουμε. Αυτό επιτυγχάνεται, τροποποιώντας τις μεταβλητές στο υπολογιστικό γράφημα. Το TensorFlow γνωρίζει πώς να τροποποιεί τις μεταβλητές επειδή παρακολουθεί τους υπολογισμούς στο μοντέλο και υπολογίζει αυτόματα τις κλίσεις της μεταβλητής, για να ελαχιστοποιήσει τη συνάρτηση απώλειας.

2.3.2.11 Δήλωση μεταβλητών και δημιουργία Tensors

Αρχικά δημιουργούμε τανυστές - tensors με τέσσερις κυρίως τρόπους :

1. Tensors σταθερού μεγέθους

   Δημιουργία μηδενικού τανυστή 3X3

   ```
   row_num, col_num = 3, 3

   zero_tsr = tf.zeros(shape=[row_num, col_num], dtype=tf.float32)
   ```

   Δημιουργούμε έναν παρόμοιο τανυστή με μονάδες(1).

   ```
   ones_tsr = tf.ones([row_num, col_num])
   ```

   Δημιουργούμε έναν σταθερό γεμάτο με τον αριθμό “23”, τανυστή.

   ```
   filled_tsr = tf.fill([row_num, col_num], 23)
   ```

   Δημιουργία τανυστή από μία υπάρχουσα σταθερά.

   ```
   constant_tsr = tf.constant([1,2,3])
   ```

Η tf.constant() συνάρτηση μπορεί να χρησιμοποιηθεί για τη μετάδοση μίας τιμής σε πίνακα που μιμείται τη συμπεριφορά tf.fill() γράφοντας:

```
tf.constant(23,[row_num, col_num]).
```

2. Tensors παρόμοιου σχήματος : Μπορούμε αρχικοποιήσουμε μεταβλητές με βάση το σχήμα άλλων tensors, ως εξής:

zeros_similar = tf.zeros_like(constant_tsr) ones_similar = tf.ones_like(constant_ts

3. Ακολουθία τανυστών: Στο TensorFlow, όλοι οι παράμετροι τεκμηριώνονται ως tensors. Το TensorFlow μας επιτρέπει να καθορίσουμε τανυστές που περιέχουν καθορισμένα διαστήματα. Οι ακόλουθες συναρτήσεις συμπεριφέρονται πολύ παρόμοια με τις linspace() εξόδους και τις range() εξόδους του NumPy .

   linear_tsr = tf.linspace(start=0.0, stop=1.0, num=3)

Ο τανυστής που προκύπτει έχει μία ακολουθία [0,0, 0,5, 1,0] (η print(linear_tsr) εντολή, θα παρέχει την απαραίτητη έξοδο). Η συνάρτηση περιλαμβάνει την καθορισμένη τιμή διακοπής. Δεν συμβαίνει το ίδιο με την tf.range συνάρτηση.

integer_seq_tsr = tf.range(start=6, limit=15, delta=3)

Το αποτέλεσμα είναι η ακολουθία [6, 9, 12], όπου η συνάρτηση δεν περιλαμβάνει την οριακή τιμή και μπορεί λειτουργεί τόσο με ακέραιες όσο και με τιμές δεκαδικές(float) για τις παραμέτρους έναρξης και ορίου.

4. Τυχαίοι τανυστές : Οι ακόλουθοι τυχαίοι αριθμοί προέρχονται από ομοιόμορφη κατανομή:

   randunif_tsr = tf.random.uniform([row_num, col_num],minval=0, maxval=1)

Η ομοιόμορφη κατανομή αντλεί από το διάστημα που περιλαμβάνει minval = 0 αλλά όχι ως και maxval = 1. Επομένως, σε αυτήν την περίπτωση, το εύρος εξόδου είναι [0, 1). Για να λάβουμε έναν τανυστή με τυχαίους αριθμούς από μία κανονική κατανομή:

randnorm_tsr = tf.random.normal([row_num, col_num],mean=0.0, stddev=1.0)

Υπάρχουν επίσης στιγμές που θέλουμε να δημιουργήσουμε κανονικές τυχαίες τιμές που αντλούνται σε ορισμένα όρια. Η truncated_normal() συνάρτηση επιλέγει πάντα τις κανονικές τιμές εντός δύο τυπικών αποκλίσεων της καθορισμένης μέσης τιμής:

runcnorm_tsr =

tf.random.truncated_normal([row_num, col_num],mean=0.0, stddev=1.0)

2.3.2.12 Πρόθυμη Εκτέλεση - Eager Execution

Στο TensorFlow 2.x, η ομάδα των προγραμματιστών του έχει προτιμήσει ως προεπιλογή μία πειραματική προσέγγιση που επιτρέπει την άμεση αξιολόγηση των λειτουργιών, διευκολύνοντας τον εντοπισμό σφαλμάτων και τη δοκιμή παραλλαγών δικτύου. Ονομάστηκε eager execution (πρόθυμη εκτέλεση) και κατά την εκτέλεση, οι πράξεις επιστρέφουν συγκεκριμένες τιμές αντί για δείκτες σε μέρη ενός υπολογιστικού γραφήματος, το οποίο θα δημιουργηθεί αργότερα. Δεν χρειάζεται να κάνουμε τίποτα, η πρόθυμη εκτέλεση είναι ο προεπιλεγμένος τρόπος λειτουργίας στο TensorFlow 2.x.

tf.executing_eagerly()

Καθώς το TensorFlow έχει πλέον ρυθμιστεί ως πρόθυμη εκτέλεση, το **tf.Session** έχει αφαιρεθεί από το TensorFlow API και δεν χρειάζεται πλέον να δημιουργήσουμε ένα υπολογιστικό γράφημα πριν εκτελέσουμε έναν υπολογισμό. Το μόνο που έχουμε να κάνουμε, είναι να δημιουργήσουμε το δίκτυό μας και να το δοκιμάσουμε στην πορεία. Αυτό ανοίγει το δρόμο για κοινές βέλτιστες πρακτικές λογισμικού, όπως η τεκμηρίωση του κώδικα, η χρήση αντικειμενοστραφούς προγραμματισμού κατά τη δέσμη ενεργειών του κώδικα και η οργάνωση του σε επαναχρησιμοποιήσιμες αυτόνομες ενότητες.

2.3.2.13 Εργασίες με πίνακες

Πολλοί αλγόριθμοι εκτελούνται με τη βοήθεια πράξεων πινάκων. Το TensorFlow μας δίνει εύχρηστες λειτουργίες για την εκτέλεση τέτοιων υπολογισμών.

Δημιουργία πινάκων : Μπορούμε να δημιουργήσουμε δισδιάστατους πίνακες από πίνακες NumPy ή ένθετες λίστες. Μπορούμε να χρησιμοποιήσουμε τις λειτουργίες δημιουργίας τανυστή και να καθορίσουμε ένα δισδιάστατο σχήμα για λειτουργίες όπως zeros(), ones(), και truncated_normal(). Το TensorFlow μας επιτρέπει επίσης να δημιουργήσουμε μία διαγώνιο πίνακα από έναν μονοδιάστατο ή λίστα, χρησιμοποιώντας τη diag() συνάρτηση, ως εξής:

```
import tensorflow as tf

import numpy as np
```

```
ident_mat=tf.linalg.diag([1.0,1.0,1.0])

A=tf.random.truncated_normal([2,3])

B=tf.fill([2,3],5.0)

C=tf.random.uniform([3,2])

D=tf.convert_to_tensor(np.array([[1.,2.,3.],[-3.,-7.,-1.],[0,5.,-2.]]),dtype=tf.float32)

print(ident_mat)
```

```
[[1.0.0.]
 [0.1.0.]
 [0.0.1.]]
```

```
print(A)
```

```
[[-0.37435472 -0.24152572 -1.1346352 ]
 [-1.3431 0.05264652 -1.4435557 ]]
```

```
print(B)
```

```
[[5. 5. 5.]
 [5. 5. 5.]]
```

```
print(C)
```

```
[[0.81086075 0.23591745]
[0.33844995 0.04469907]
[0.6694279 0.6594739 ]]
```

```
print(D)
```

```
[[ 1. 2. 3.]
[-3. -7. -1.]
[ 0. 5. -2.]]
```

```
print(A+B)
```

[[4.625645 4.7584743 3.8653648]

[3.6569 5.0526466 3.5564442]]

print(tf.matmul(B,ident_mat))

[[5. 5. 5.]

[5. 5. 5.]]

Οι tensors A,C, δημιουργούνται με έναν τυχαίο τρόπο και πιθανότατα θα διαφέρουν στην επανεκτέλεση.

Πρόσθεση, αφαίρεση και πολλαπλασιασμός : Για να προσθέσουμε ή να αφαιρέσουμε πίνακες της ίδιας διάστασης, TensorFlow χρησιμοποιεί τις συνάρτησεις:

print(A+B) , print(A-B)

Για τον πολλαπλασιασμό :

print(tf.matmul(B, ident_mat))

Το tensorflow μπορεί να εκτελέσει πολλαπλασιασμό μεταξύ δύο πινάκων του ίδιου σχήματος και τύπου χρησιμοποιώντας την tf.multiply συνάρτηση:

print(tf.multiply(D, ident_mat))

Η διαίρεση πίνακα δεν ορίζεται ρητά και πολλοί ορίζουν τη διαίρεση, πολλαπλασιάζοντας με το αντίστροφο. Είναι ουσιαστικά διαφορετική από τη διαίρεση με τον πραγματικό αριθμό.

Μεταφορά : Μεταφέρουμε έναν πίνακα (αναστρέφουμε τις στήλες και τις γραμμές) ως εξής:

print(tf.transpose(C))

Αντιστροφή πίνακα : Για να βρούμε τον αντίστροφο ενός τετραγωνικού πίνακα :

print(tf.linalg.inv(D))

Ιδιοτιμές και ιδιοδιανύσματα : Για ιδιοτιμές και ιδιοδιανύσματα, χρησιμοποιούμε τον ακόλουθο κώδικα:

```
print(tf.linalg.eigh(D))
```

Η βιβλιοθήκη TensorFlow έχει τις τυπικές λειτουργίες για τανυστές, που είναι, **add(), subtract(), multiply() και division()**.

Μία άλλη σημαντική συνάρτηση είναι η mod(), η οποία επιστρέφει το υπόλοιπο της διαίρεσης ως εξής:

```
print(tf.math.mod(22.0, 5.0))
```

tf.Tensor(2.0, shape=(), dtype=float32)

Ακολουθεί μία λίστα με βασικές μαθηματικές συναρτήσεις στο TensorFlow. Όλες αυτές οι συναρτήσεις, λειτουργούν κατά στοιχείο (elementwise).

tf.math.abs()	Απόλυτη τιμή ενός τανυστή εισόδου
tf.math.ceil()	Λειτουργία οροφής ενός τανυστή εισόδου
tf.math.cos()	Συνάρτηση συνημίτονου ενός τανυστή εισόδου
tf.math.exp()	Βάση e, εκθετικό ενός τανυστή εισόδου
tf.math.floor()	Λειτουργία floor ενός τανυστή εισόδου
tf.linalg.inv()	Πολλαπλασιαστικός αντίστροφος(1/x) ενός tensor εισόδου
tf.math.log()	Φυσικός λογάριθμος ενός τανυστή εισόδου
tf.math.maximum()	Επιστρέφει το max από δύο τανυστές (elementwise)
tf.math.minimum()	Επιστρέφει το min από δύο τανυστές (elementwise)
tf.math.negative()	Αρνητικό ενός τανυστή εισόδου

tf.math.pow()	Ο 1ος τανυστής ανυψώθηκε στο 2ο τανυστή elementwise
tf.math.round()	Στρογγυλοποίηση στον πλησιέστερο ακέραιο
tf.math.rsqrt()	Το αντίστροφο της τετραγωνικής ρίζας ενός τανυστή
tf.math.sign()	Επιστρέφει -1, 0 ή 1, ανάλογα με το πρόσημο του τανυστή
tf.math.sin()	Λειτουργία ημιτόνου ενός τανυστή εισόδου
tf.math.sqrt()	Τετραγωνική ρίζα ενός τανυστή εισόδου
tf.math.square()	Τετράγωνο ενός τανυστή εισόδου

Εξειδικευμένες μαθηματικές συναρτήσεις :

Υπάρχουν μερικές ειδικές μαθηματικές συναρτήσεις που χρησιμοποιούνται συχνά στη μηχανική εκμάθηση και το TensorFlow έχει ενσωματωμένες λειτουργίες για αυτές.

tf.math.digamma(), Συνάρτηση Psi, παράγωγο της lgamma() συνάρτησης

tf.math.erf(), Συνάρτηση σφάλματος Gauss, στοιχείο προς στοιχείο ενός τανυστή

tf.math.erfc(), Συμπληρωματική συνάρτηση σφάλματος ενός τανυστή

tf.math.igamma(), Χαμηλότερη κανονικοποιημένη ατελής λειτουργία γάμμα

tf.math.igammac(), Ανώτερη κανονικοποιημένη ατελής συνάρτηση γάμμα

tf.math.lbeta(), Φυσικός λογάριθμος της απόλυτης τιμής της συνάρτησης βήτα

tf.math.lgamma(), Φυσικός λογάριθμος της απόλυτης τιμής της συνάρτησης γάμμα

tf.math.squared_difference(), Υπολογίζει το τετράγωνο των διαφορών μεταξύ 2 τανυστών

Μπορούμε επίσης να δημιουργήσουμε πολλές διαφορετικές προσαρμοσμένες συναρτήσεις ως συνθέσεις των προηγούμενων, ως εξής:

Tangent function (tan(pi/4)=1)

```
def pi_tan(x):
    return tf.tan(3.1416/x)
print(pi_tan(4))
```

tf.Tensor(1.0000036, shape=(), dtype=float32)

2.3.2.14 Λειτουργίες Ενεργοποίησης

Οι λειτουργίες ενεργοποίησης επιτελούν σημαντικό ρόλο στα νευρωνικά δίκτυα, στην προσέγγιση μη γραμμικών εξόδων και στην προσαρμογή σε μη γραμμικά χαρακτηριστικά. Οι συναρτήσεις ενεργοποίησης αποτελούν ουσιαστικό μέρος οποιουδήποτε νευρωνικού δικτύου, εστιάζοντας στην προσαρμογή των βαρών και των δυναμικών πόλωσης(bias-threshold). Στο TensorFlow, οι συναρτήσεις ενεργοποίησης είναι μη γραμμικές λειτουργίες που δρουν σε tensors. Η κύρια ιδέα είναι ότι εισάγουν μία μη γραμμικότητα στο γράφημα, ενώ εξομαλύνουν τις εξόδους.

Οι συναρτήσεις ενεργοποίησης έχουν ενσωματωθεί στη βιβλιοθήκη των νευρωνικών δικτύων (nn) στο TensorFlow. Εκτός από τη χρήση ενσωματωμένων λειτουργιών ενεργοποίησης, έχουμε τη δυνατότητα να σχεδιάσουμε τις δικές μας, χρησιμοποιώντας τις λειτουργίες του TensorFlow. Μπορούμε να εισάγουμε τις προκαθορισμένες συναρτήσεις ενεργοποίησης (από **tensorflow import nn**) Η διορθωμένη γραμμική μονάδα, γνωστή ως ReLU, είναι ο πιο συνηθισμένος και βασικός τρόπος εισαγωγής της μη γραμμικότητας στα νευρωνικά δίκτυα. Αυτή η συνάρτηση απεικονίζεται ως max(0,x). Είναι συνεχής, αλλά όχι ομαλή. Εμφανίζεται ως εξής:

```
print(tf.nn.relu([-3., 3., 10.]))
```

tf.Tensor([0. 3. 10.], shape=(3,), dtype=float32)

Υπάρχουν στιγμές που θα θέλουμε να περιορίσουμε το γραμμικά αυξανόμενο τμήμα της συνάρτησης ενεργοποίησης ReLU. Μπορούμε να το κάνουμε αυτό τοποθετώντας τη max(0,x) συνάρτηση σε μία min() συνάρτηση. Η εφαρμογή που έχει το TensorFlow ονομάζεται συνάρτηση ReLU6. Αυτό ορίζεται ως **min(max(0,x),6)**. Αυτή είναι μία έκδοση της σιγμοειδούς συνάρτησης, είναι υπολογιστικά γρηγορότερη και δεν υποφέρει από εξαφάνιση (απείρως κοντά στο μηδέν) ή από εκρήξεις τιμών. Εμφανίζεται ως εξής:

print(tf.nn.relu6([-3., 3., 10.]))

tf.Tensor([0. 3. 6.], shape=(3,), dtype=float32)

Η σιγμοειδής συνάρτηση είναι η πιο κοινή συνεχής και ομαλή συνάρτηση ενεργοποίησης. Ονομάζεται επίσης συνάρτηση logistics και έχει τη μορφή 1 / (1 + exp (-x)) . Η σιγμοειδής συνάρτηση δεν χρησιμοποιείται πολύ συχνά λόγω της τάσης της να μηδενίζει τους όρους του backpropagation κατά τη διάρκεια της εκπαίδευσης του δικτύου. Εμφανίζεται ως εξής:

print(tf.nn.sigmoid([-1., 0., 1.]))

tf.Tensor([0.26894143 0.5 0.7310586], shape=(3,), dtype=float32)

Ορισμένες συναρτήσεις ενεργοποίησης, όπως η σιγμοειδής, δεν κινούνται πάντα γύρω από το μηδέν. Για να το αποφύγουμε, θα ήταν σκόπιμο να αρχικοποιούμε στο μηδέν τα δεδομένα, πριν τα χρησιμοποιήσουμε στους περισσότερους αλγόριθμους υπολογιστικών γραφημάτων.

Μία άλλη ομαλή λειτουργία ενεργοποίησης είναι η υπερβολική εφαπτομένη. Η συνάρτηση της μοιάζει πολύ με τη σιγμοειδής, εκτός από το ότι αντί να έχει εύρος μεταξύ 0 και 1, έχει εύρος μεταξύ -1 και 1. Αυτή η συνάρτηση έχει τη μορφή της αναλογίας του υπερβολικού ημιτόνου προς το υπερβολικό συνημίτονο. Ο τύπος της είναι : **(**(exp(x) – exp(-x)**) / (**exp(x) + exp(-x)**).**

Η λειτουργία ενεργοποίησης διατυπώνεται :

print(tf.nn.tanh([-1., 0., 1.]))

tf.Tensor([-0.7615942 0. 0.7615942], shape=(3,), dtype=float32)

Η softsign συνάρτηση χρησιμοποιείται επίσης ως συνάρτηση ενεργοποίησης. Η μορφή αυτής της συνάρτησης είναι x/(| x | + 1) . Η softsign συνάρτηση υποτίθεται ότι είναι μία συνεχής (αλλά όχι ομαλή) προσέγγιση της συνάρτησης σημείου. Ο κώδικας γράφεται :

print(tf.nn.softsign([-1., 0., -1.]))

tf.Tensor([-0.5 0. -0.5], shape=(3,), dtype=float32)

Μία άλλη λειτουργία, η softplus συνάρτηση, είναι μία ομαλή έκδοση της συνάρτησης ReLU. Ο τύπος αυτής της συνάρτησης είναι log (exp (x) + 1) . Εμφανίζεται ως εξής:

print(tf.nn.softplus([-1., 0., -1.]))

tf.Tensor([0.31326166 0.6931472 0.31326166], shape=(3,), dtype=float32)

Η softplus συνάρτηση πηγαίνει στο άπειρο καθώς η είσοδος αυξάνεται, ενώ η softsign συνάρτηση πηγαίνει στο 1. Όταν η είσοδος γίνεται μικρότερη, η softplus συνάρτηση πλησιάζει στο μηδέν και η softsign συνάρτηση πηγαίνει στο -1.

Εκτός από τις ανωτέρω συναρτήσεις ενεργοποίησης, μπορούμε να βρούμε ένα πληρέστερο κατάλογο συναρτήσεων, στον ιστότοπο του Tensorflow στο Keras στην ενότητα ενεργοποίησης, διεύθυνση : https://www.tensorflow.org/api_docs/python/tf/keras/activations

Οι συναρτήσεις ενεργοποίησης είναι η μέθοδος με την οποία μπορούμε να εισάγουμε μη γραμμικότητα σε νευρωνικά δίκτυα ή άλλα υπολογιστικά γραφήματα στο μέλλον. Είναι σημαντικό να σημειωθεί, πού στο δίκτυό μας χρησιμοποιούμε λειτουργίες ενεργοποίησης. Εάν η συνάρτηση ενεργοποίησης έχει εύρος μεταξύ 0 και 1 (σιγμοειδής), τότε το υπολογιστικό γράφημα μπορεί να εξάγει μόνο τιμές μεταξύ 0 και 1. Εάν οι συναρτήσεις ενεργοποίησης είναι κρυμμένες μεταξύ κόμβων, τότε θέλουμε να γνωρίζουμε την επίδραση που έχει το εύρος των τιμών στους tensors καθώς τους περνάμε. Εάν οι τανυστές μας έχουν κλιμακωθεί ώστε να έχουν μέσο μηδέν, θα θέλουμε να χρησιμοποιήσουμε μία συνάρτηση ενεργοποίησης που διατηρεί όσο το δυνατόν μεγαλύτερη διακύμανση γύρω στο μηδέν. Θα μπορούσαμε να επιλέξουμε μία λειτουργία ενεργοποίησης όπως η υπερβολική εφαπτομένη (tanh) ή το softsign . Αν οι tensors έχουν σχεδιαστεί να είναι θετικοί, θα επιλέγαμε ιδανικά μία συνάρτηση ενεργοποίησης, που διατηρεί τη διακύμανση στον θετικό άξονα.

Υπάρχει η δυνατότητα να δημιουργήσουμε προσαρμοσμένες ενεργοποιήσεις όπως το Swish, το οποίο είναι x *sigmoid (x) (βλ. Swish: a Self-Gated Activation Function , Ramachandran et al., 2017, https://arxiv.org/abs/1710.05941), το οποίο μπορεί να χρησιμοποιηθεί ως μία πιο αποτελεσματική αντικατάσταση για ενεργοποιήσεις ReLU, σε προβλήματα εικόνας και πίνακα δεδομένων:

```
def swish(x):
    return x * tf.nn.sigmoid(x)

print(swish([-1., 0., 1.]))
```

tf.Tensor([-0.26894143 0. 0.7310586], shape=(3,), dtype=float32)

2.4 Πηγές Δεδομένων (data sources)

Η προσαρμογή αλγορίθμων μηχανικής μάθησης, προϋποθέτει την άντληση μεγάλου όγκου δεδομένων, για την εκπαίδευση και την αξιολόγηση των μοντέλων μας.

Στη βιβλιοθήκη TensorFlow (TFDS), υπάρχει όγκος δεδομένων, τα οποία αποτελούν μία μεγάλη συλλογή από σύνολα δεδομένων έτοιμα προς χρήση (ενημέρωση για την πλήρη λίστα εδώ: https://www.tensorflow.org/datasets/catalog/overview).

Για την εγκατάσταση των TFDS, εκτελούμε την ακόλουθη εντολή στην κονσόλα μας:

```
pip install tensorflow-datasets
```

Ορισμένες από τις πηγές δεδομένων βασίζονται σε εξωτερικούς ιστότοπους, όπως:

Δεδομένα ίριδας (Iris data): Το κλασικό δομημένο σύνολο δεδομένων που χρησιμοποιείται στη μηχανική μάθηση και ίσως σε όλα τα παραδείγματα στατιστικών. Είναι ένα σύνολο δεδομένων που μετρά το μήκος από τον κάλυκα του άνθους, το πλάτος του, το μήκος των πετάλων και το πλάτος των πετάλων, τριών διαφορετικών τύπων λουλουδιών ίριδας: Iris setosa, Iris virginica και Iris versicolor. Υπάρχουν συνολικά 150 μετρήσεις, πράγμα που σημαίνει ότι υπάρχουν 50 μετρήσεις για κάθε είδος. Για να φορτώσουμε το σύνολο δεδομένων στην Python, θα χρησιμοποιήσουμε συναρτήσεις TFDS, ως εξής:

```
import tensorflow_datasets as tfds

iris = tfds.load('iris', split='train')
```

Δεδομένα βάρους γέννησης (Birth weight data) : Τα δεδομένα ήταν αρχικά από το Ιατρικό Κέντρο Baystate, Springfield, Mass, 1986. Αυτό το σύνολο δεδομένων περιέχει μετρήσεις που περιλαμβάνουν το βάρος του νεογέννητου και άλλες δημογραφικές και ιατρικές μετρήσεις της μητέρας και του οικογενειακού ιστορικού. Υπάρχουν 189 παρατηρήσεις έντεκα μεταβλητών.

Δεδομένα στέγασης της Βοστώνης : Το Πανεπιστήμιο Carnegie Mellon διατηρεί μία βιβλιοθήκη με σύνολα δεδομένων στη StatLibΒιβλιοθήκη τους. Αυτά τα δεδομένα είναι εύκολα προσβάσιμα μέσω του Πανεπιστημίου της Καλιφόρνια στο αποθετήριο μηχανικής μάθησης του Irvine (https://archive.ics.uci.edu/ml/index.php). Υπάρχουν 506 παρατηρήσεις της αξίας ενός σπιτιού, μαζί με διάφορα δημογραφικά δεδομένα και χαρακτηριστικά κατοικίας (14 μεταβλητές). Ο ακόλουθος κώδικας δείχνει πώς αποκτάται η πρόσβαση σε αυτά τα δεδομένα στο TensorFlow:

```
import tensorflow_datasets as tfds

housing_url =

'http://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.data'

path = tf.keras.utils.get_file(housing_url.split("/")[-1], housing_url)

def map_line(x):

    return tf.strings.to_number(tf.strings.split(x))

housing = (tf.data.TextLineDataset(path).map(map_line))
```

Στοιχεία χειρόγραφου MNIST : Το σύνολο δεδομένων του Mixed National Institute of Standards and Technology (MNIST) είναι υποσύνολο της μεγαλύτερης βάσης δεδομένων χειρογράφων NIST. Το σύνολο δεδομένων χειρόγραφου MNIST φιλοξενείται στον ιστότοπο του Yann LeCun (http://yann.lecun.com/exdb/mnist/). Είναι μία βάση δεδομένων με 70.000 εικόνες μονοψήφιων αριθμών (0-9), με περίπου 60.000 εικόνες για εκπαίδευση και 10.000 για ένα σύνολο δοκιμών. Αυτό το σύνολο δεδομένων χρησιμοποιείται τόσο συχνά στην αναγνώριση εικόνας που το TensorFlow παρέχει ενσωματωμένες λειτουργίες για πρόσβαση σε αυτά τα δεδομένα. Στη μηχανική μάθηση είναι επίσης σημαντικά τα δεδομένα επικύρωσης για την αποφυγή υπερβολικής προσαρμογής (overfitting). Ο ακόλουθος κώδικας επιτρέπει την πρόσβαση σε αυτά τα δεδομένα στο TensorFlow:

```
import tensorflow_datasets as tfds

mnist = tfds.load('mnist', split=None)

mnist_train = mnist['train']

mnist_test = mnist['test']
```

Δεδομένα κειμένου ανεπιθύμητης αλληλογραφίας (Spam-ham text data) : Το αποθετήριο δεδομένων μηχανικής μάθησης UCI, περιέχει επίσης ένα σύνολο δεδομένων μηνυμάτων κειμένου spam-ham. Μπορούμε να έχουμε πρόσβαση σε αυτό το .zip αρχείο και να λάβουμε τα δεδομένα κειμένου spam-ham ως εξής:

```
import tensorflow_datasets as tfds

zip_url =

'http://archive.ics.uci.edu/ml/machine-learning-databases/00228/smsspamcollection.zip'

path =

tf.keras.utils.get_file(zip_url.split("/")[-1], zip_url, extract=True)

path =

path.replace("smsspamcollection.zip", "SMSSpamCollection")

def split_text(x):

    return tf.strings.split(x, sep='\t')

text_data = (tf.data.TextLineDataset(path).map(split_text))
```

Δεδομένα κριτικής ταινίας : Ο Bo Pang από το πανεπιστήμιο Cornell, κυκλοφόρησε ένα σύνολο δεδομένων από κριτικές ταινιών, που ταξινομεί τις κριτικές σε καλές ή κακές.

Υπάρχει η δυνατότητα να βρούμε αυτά τα δεδομένα στον ιστότοπο του Πανεπιστημίου Cornell: http://www.cs.cornell.edu/people/pabo/movie-review-data/. Για λήψη, εξαγωγή και μετατροπή αυτών των δεδομένων, μπορούμε να εκτελέσουμε τον ακόλουθο κώδικα:

```
import tensorflow_datasets as tfds

movie_data_url =

'http://www.cs.cornell.edu/people/pabo/movie-review-data/rt-polaritydata.tar.gz'

path = tf.keras.utils.get_file(movie_data_url.split("/")[-1], movie_data_url, extract=True)

path = path.replace('.tar.gz', '')

with open(path+filename, 'r', encoding='utf-8', errors='ignore') as movie_file:

    for response, filename in enumerate(['\\rt-polarity.neg', '\\rt-polarity.pos']):

        with open(path+filename, 'r') as movie_file:
```

```
for line in movie_file: review_file.write(str(response) + '\t' + line.encode('utf-8').decode())

def split_text(x):

    return tf.strings.split(x, sep='\t')

movies = (tf.data.TextLineDataset('movie_reviews.txt').map(split_text))
```

Δεδομένα εικόνας CIFAR-10 (CIFAR-10 image data) : Το Καναδικό Ινστιτούτο Προηγμένης Έρευνας κυκλοφόρησε ένα σύνολο εικόνων που περιέχει 80 εκατομμύρια έγχρωμες εικόνες (κάθε εικόνα είναι μεγέθους 32 x 32 εικονοστοιχείων). Υπάρχουν 10 διαφορετικές κατηγορίες (αεροπλάνο, αυτοκίνητο, πουλί και ούτω καθεξής). Το CIFAR-10 είναι ένα υποσύνολο που περιλαμβάνει 60.000 εικόνες. Υπάρχουν 50.000 εικόνες στο σετ εκπαίδευσης και 10.000 στο σετ δοκιμών. Για να κατεβάσουμε αυτό το σύνολο δεδομένων, εκτελούμε τον ακόλουθο κώδικα :

```
import tensorflow_datasets as tfds

ds, info = tfds.load('cifar10', shuffle_files=True, with_info=True)

print(info)

cifar_train = ds['train']

cifar_test = ds['test']
```

Λίστα πόρων του TensorFlow για περαιτέρω εκμάθηση:

Η επίσημη τεκμηρίωση TensorFlow Python API βρίσκεται στη διεύθυνση https://www.tensorflow.org/api_docs/python . Εδώ, υπάρχει τεκμηρίωση και παραδείγματα όλων των συναρτήσεων, αντικειμένων και μεθόδων στο TensorFlow.

Τα επίσημα μαθήματα του TensorFlow είναι πολύ διεξοδικά και λεπτομερή. Βρίσκονται στη διεύθυνση https://www.tensorflow.org/tutorials/index.html . Αρχίζουν να καλύπτουν μοντέλα αναγνώρισης εικόνας και λειτουργούν μέσω των μοντέλων Word2Vec, RNN. Επίσης προσθέτουν συνεχώς περισσότερα σεμινάρια και παραδείγματα.

Το επίσημο αποθετήριο GitHub του TensorFlow είναι διαθέσιμο στη διεύθυνση https://github.com/tensorflow/tensorflow.

Ένα δημόσιο κοντέινερ Docker που ενημερώνεται από το TensorFlow, είναι διαθέσιμο στο Dockerhub στη διεύθυνση https://hub.docker.com/r/tensorflow/tensorflow/

Μία μεγάλη πηγή άντλησης βοήθειας είναι το Stack Overflow. Υπάρχει μία ετικέτα για το TensorFlow. Για να δείτε τη δραστηριότητα σε αυτήν την ετικέτα, επισκεφτείτε τη διεύθυνση http://stackoverflow.com/questions/tagged/Tensorflow

Ενώ το TensorFlow είναι πολύ ευέλικτο και μπορεί να χρησιμοποιηθεί για πολλά πράγματα, η πιο κοινή χρήση του TensorFlow είναι η βαθιά μάθηση (DL). Για να γίνει κατανοητή η βάση της βαθιάς μάθησης, το πώς λειτουργούν τα υποκείμενα μαθηματικά και να αναπτύξει περισσότερη διαίσθηση στη βαθιά μάθηση, η Google δημιούργησε ένα διαδικτυακό μάθημα που είναι διαθέσιμο στο Udacity.

ΚΕΦΑΛΑΙΟ 3ο

Υλοποίηση μοντέλων για ταξινόμηση εικόνων και επεξεργασία φυσικής γλώσσας

Υλοποιούνται δύο μοντέλα ταξινόμησης εικόνων (image classification) κι ένα μοντέλο επεξεργασίας φυσικής γλώσσας (Natural Language Processing). Τα μοντέλα ταξινόμησης εικόνων τροφοδοτούνται από το σύνολο δεδομένων mnist και αναπτύσσονται με δύο αρχιτεκτονικές νευρωνικών δικτύων. Το πρώτο υλοποιείται με CNN και το δεύτερο με LSTM αρχιτεκτονική δικτύου. Το μοντέλο NLP, αφορά την ανάλυση συναισθήματος και τροφοδοτείται από το αρχείο imdb.

3.1 Πεδία εφαρμογής όρασης υπολογιστών

Η όραση των υπολογιστών ως τομέας ξεκίνησε ήδη από τη δεκαετία του '60, μεταξύ της ερευνητικής κοινότητας της Τεχνητής Νοημοσύνης (AI). Ο Μάρβιν Μίνσκι (Marvin Minsky) ήταν ένας από τους πρώτους που περιέγραψε μία προσέγγιση για την κατασκευή συστημάτων τεχνητής νοημοσύνης με βάση την αντίληψη (*Steps Toward Artificial intelligence , Proceedings of the IRE, 1961*). Υποστήριξε ότι με τη χρήση χαμηλότερων λειτουργιών όπως η αναγνώριση προτύπων, η μάθηση, ο προγραμματισμός και η επαγωγή, θα μπορούσε να είναι δυνατή η κατασκευή μηχανών ικανών να λύσουν μία ευρεία ποικιλία προβλημάτων. Αυτή η θεωρία διερευνήθηκε σωστά μόνο από τη δεκαετία του '80 και μετά. Στο Locomotion, Vision, and Intelligence το 1984, ο Hans Moravec σημείωσε ότι το νευρικό μας σύστημα, μέσω της διαδικασίας της εξέλιξης, έχει αναπτυχθεί για να αντιμετωπίζει αντιληπτικές εργασίες (περισσότερο από το 30% του εγκεφάλου μας είναι αφιερωμένο στην όραση!). Όπως σημείωσε, ακόμα κι αν οι υπολογιστές είναι αρκετά καλοί στην αριθμητική, δεν μπορούν να ανταγωνιστούν τις αντιληπτικές μας ικανότητες. Εμπνευσμένοι από την ανθρώπινη αντίληψη, οι βασικοί μηχανισμοί της όρασης υπολογιστή δεν έχουν εξελιχθεί πολύ από τα πρώτα χρόνια. Η ιδέα είναι πρώτα να εξαχθούν σημαντικά χαρακτηριστικά από τα ακατέργαστα pixel και στη συνέχεια να αντιστοιχιστούν αυτά τα χαρακτηριστικά με ετικέτα, προκειμένου να επιτευχθεί αναγνώριση. Κάθε χαρακτηριστικό είναι ένα κομμάτι πληροφορίας (συχνά αναπαρίσταται μαθηματικά ως μονοδιάστατο ή δισδιάστατο διάνυσμα) που εξάγεται από δεδομένα που σχετίζονται με την εκάστοτε εργασία. Τα χαρακτηριστικά περιλαμβάνουν ορισμένα βασικά

σημεία στις εικόνες, συγκεκριμένα άκρα, διακριτικές ενημερώσεις κ.λπ. Η ικανότητά μας να αποκρυπτογραφούμε τα οπτικά ερεθίσματα που συλλαμβάνουν τα μάτια μας, να ξεχωρίζουμε αμέσως ένα αντικείμενο από το άλλο ή να αναγνωρίζουμε το πρόσωπο κάποιου που έχουμε συναντήσει μόνο μία φορά, είναι μοναδική. Για τις μηχανές, οι εικόνες είναι απλώς κηλίδες pixel, πίνακες τιμών κόκκινου-πράσινου-μπλε και με βάση αυτά τα δεδομένα εξάγονται ουσιαστικές πληροφορίες (όπως τα αντικείμενα που υπάρχουν στις εικόνες, η θέση τους και ο αριθμό τους). Αυτό το γενικό πρόβλημα μπορεί να χωριστεί σε πολλούς υποτομείς όπως:

Ταξινόμηση αντικειμένων : Η ταξινόμηση αντικειμένων (ή ταξινόμηση εικόνων) είναι η εργασία της ανάθεσης κατάλληλων ετικετών (ή κλάσεων) σε εικόνες.

Αναγνώριση αντικειμένου : Ενώ οι μέθοδοι ταξινόμησης αντικειμένων εκχωρούν ετικέτες από ένα προκαθορισμένο σύνολο, οι μέθοδοι αναγνώρισης αντικειμένων (ή ταξινόμησης παρουσιών) μαθαίνουν να αναγνωρίζουν συγκεκριμένες παρουσίες μίας κλάσης. Για παράδειγμα, ένα εργαλείο ταξινόμησης αντικειμένων θα μπορούσε να διαμορφωθεί ώστε να επιστρέφει εικόνες που περιέχουν πρόσωπα, ενώ μία μέθοδος αναγνώρισης θα επικεντρωνόταν στα χαρακτηριστικά του προσώπου για την αναγνώριση του ατόμου και την αναγνώριση τους σε άλλες εικόνες (αναγνώριση κάθε προσώπου σε όλες τις εικόνες). Η αναγνώριση αντικειμένων μπορεί να θεωρηθεί ως μία διαδικασία για την ομαδοποίηση ενός συνόλου δεδομένων.

Ανίχνευση και εντοπισμός αντικειμένων : Μία άλλη εργασία είναι η ανίχνευση συγκεκριμένων στοιχείων σε μία εικόνα. Εφαρμόζεται συνήθως για την ανίχνευση προσώπου, για εφαρμογές επιτήρησης ή ακόμα και για προηγμένες εφαρμογές κάμερας, την ανίχνευση καρκινικών κυττάρων στην ιατρική, την ανίχνευση κατεστραμμένων εξαρτημάτων σε βιομηχανικές εγκαταστάσεις κ.λπ.

Τμηματοποίηση αντικειμένων και παρουσιών : Η τμηματοποίηση μπορεί να θεωρηθεί ως ένας πιο προηγμένος τύπος ανίχνευσης. Αντί να παρέχουν απλώς οριοθετημένα πλαίσια για τα αναγνωρισμένα στοιχεία, οι μέθοδοι τμηματοποίησης επιστρέφουν μάσκες που επισημαίνουν όλα τα εικονοστοιχεία που ανήκουν σε μία συγκεκριμένη κλάση ή σε μία συγκεκριμένη παρουσία μίας κλάσης

Εκτίμηση πόζας : Για άκαμπτα αντικείμενα, συνήθως σημαίνει την εκτίμηση των θέσεων και των προσανατολισμών των αντικειμένων σε σχέση με την κάμερα στον τρισδιάστατο χώρο. Αυτό είναι ιδιαίτερα χρήσιμο για τα ρομπότ ώστε να μπορούν να αλληλεπιδρούν με το περιβάλλον τους (επιλογή αντικειμένων, αποφυγή σύγκρουσης κ.λπ.). Για μη άκαμπτα

στοιχεία, η εκτίμηση πόζας μπορεί επίσης να σημαίνει την εκτίμηση των θέσεων των υποτμημάτων τους σε σχέση μεταξύ τους . Πιο συγκεκριμένα, όταν θεωρούνται οι άνθρωποι ως μη άκαμπτοι στόχοι, τυπικές εφαρμογές είναι η αναγνώριση ανθρώπινων στάσεων (όρθια, καθιστή, τρέξιμο, κ.λπ.).

Ανάλυση βίντεο : Η όραση υπολογιστή δεν ισχύει μόνο για μεμονωμένες εικόνες, αλλά και για βίντεο. Ο εντοπισμός συγκεκριμένων στοιχείων σε μία ροή βίντεο θα μπορούσε να γίνει καρέ προς καρέ εφαρμόζοντας μεθόδους ανίχνευσης και αναγνώρισης σε κάθε πλαίσιο. Ωστόσο, είναι πολύ πιο αποτελεσματικό να χρησιμοποιούμε προηγούμενα αποτελέσματα για τη μοντελοποίηση της κίνησης των στιγμιότυπων, προκειμένου να προβλέψουμε εν μέρει τις θέσεις τους σε μελλοντικά πλαίσια.

Αναγνώριση δράσης : Η αναγνώριση ενεργειών ανήκει στη λίστα εργασιών που μπορούν να εκτελεστούν μόνο με μία ακολουθία εικόνων. Αναγνώριση μίας δράσης σημαίνει αναγνώριση μίας συγκεκριμένης κίνησης μεταξύ ενός προκαθορισμένου συνόλου (για παράδειγμα, για ανθρώπινες ενέργειες—χορός, κολύμπι, σχεδίαση τετραγώνου ή σχεδίαση κύκλου).

Εκτίμηση κίνησης : Αντί να προσπαθούν να αναγνωρίσουν κινούμενα στοιχεία, ορισμένες μέθοδοι επικεντρώνονται στην εκτίμηση της πραγματικής ταχύτητας-τροχιάς που καταγράφεται στα βίντεο. Είναι επίσης σύνηθες να αξιολογείται η κίνηση της ίδιας της κάμερας σε σχέση με την αντιπροσωπευόμενη σκηνή (egomotion). Αυτό είναι ιδιαίτερα χρήσιμο στη βιομηχανία της ψυχαγωγίας, για παράδειγμα, για την καταγραφή της κίνησης για την εφαρμογή οπτικών εφέ ή για την επικάλυψη τρισδιάστατων πληροφοριών σε τηλεοπτικές ροές, όπως αθλητικές μεταδόσεις.

Έκδοση εικόνων με επίγνωση περιεχομένου : Εκτός από την ανάλυση του περιεχομένου τους, μέθοδοι υπολογιστικής όρασης μπορούν επίσης να εφαρμοστούν για τη βελτίωση των ίδιων των εικόνων . Όλο και περισσότερο, τα βασικά εργαλεία επεξεργασίας εικόνας αντικαθίστανται από πιο έξυπνες μεθόδους που μπορούν να χρησιμοποιούν την προηγούμενη γνώση του περιεχομένου της εικόνας για να βελτιώσουν την οπτική της ποιότητα. Για παράδειγμα, εάν μία μέθοδος μάθει πώς μοιάζει συνήθως ένα πουλί, μπορεί να εφαρμόσει αυτή τη γνώση προκειμένου να αντικαταστήσει τα θορυβώδη εικονοστοιχεία με συνεκτικά σε εικόνες πουλιών. Αυτή η ιδέα ισχύει για οποιονδήποτε τύπο αποκατάστασης εικόνας, είτε πρόκειται για αποθορυβοποίηση, αφαίρεση θολώματος ή βελτίωση ανάλυσης.

Ανακατασκευή σκηνής : Η ανακατασκευή σκηνής είναι το έργο της ανάκτησης της τρισδιάστατης γεωμετρίας μίας σκηνής, με δεδομένη μία ή περισσότερες εικόνες.

3.2 Υλοποίηση μοντέλων για ταξινόμηση εικόνας (image classification)

Στη συνέχεια θα αναπτυχθούν δύο είδη νευρωνικών δικτύων για την αναγνώριση εικόνας στο Google Colaboratoryoratory. Στο πρώτο εφαρμόζεται η αρχιτεκτονική συνελικτικού δικτύου (CNN) και στο δεύτερο αναδρομικού δικτύου (RNN-LSTM). Η τροφοδοσία τους γίνεται από την ίδια πηγή δεδομένων, το αρχείο mnist. Η πηγή αυτή περιέχει 70,000 εικόνες αριθμών με τις ετικέτες τους. Η σύγκριση δύο διαφορετικών αρχιτεκτονικών ανάπτυξης νευρωνικών δικτύων, με κοινές εισόδους, θα μας δώσει πληροφορίες σχετικά με την αποτελεσματικότητα των μοντέλων στην αναγνώριση εικόνας. Στη διαδικασία που ακολουθεί, στρώμα θεωρείται ό,τι εκπαιδεύει τις υπερπαραμέτρους κι όχι τα στρώματα συγκέντρωσης.

CNN model

```
#Εισάγουμε τις βιβλιοθήκες μας

import tensorflow as tf
from tensorflow import keras
from keras import datasets, layers, models
import numpy as np
import matplotlib.pyplot as plt

# Φορτώνουμε τα δεδομένα μας από mnist (εικόνες αριθμών σε διάσταση 28X28)

(train_images, train_labels), (test_image, test_labels) =  datasets.mnist.load_data()

# Τυπώνει τον αριθμό των δεδομένων μας για εκπαίδευση και έλεγχο

print(f"Our data contain {len(train_images)} images for training and {len(test_image)} images for testing"+"\n")

print(train_images.shape)

# Κατά την είσοδο των δεδομένων δηλώνουμε μόνο τις διαστάσεις 28X28 και το χρώμα "1"

train_images= train_images.reshape(60000, 28, 28, 1)
```

```
test_images = test_image.reshape(10000, 28, 28, 1)
# Κανονικοποιούμε τις εικόνες, ώστε να πάρουν τιμές μεταξύ 0 και 1
train_images, test_images = train_images/255.0 , test_images/255.0
# Δημιουργία του μοντέλου CNN
model = models.Sequential()
# Δηλώνω το 1ο συνελικτικό(convolutional) στρώμα με 32 φίλτρα διαστάσεων 3X3,
# με συνάρτηση ενεργοποίησης ReLu και
# διαστάσεις εικόνων εισόδου (28X28) σε γκρι δηλ 1, όχι RBG(3)
model.add(layers.Conv2D(32, (3, 3), activation = "relu", input_shape = (28, 28, 1)))
# Κάθε στρώμα μετά το 1ο, λαμβάνει ως είσοδο, την έξοδο του προηγούμενου στρώματος
# Το 1ο συνελικτικό στρώμα, ακολουθεί ένα στρώμα συγκέντρωσης(MaxPooling)
# διαστάσεων 2X2
model.add(layers.MaxPooling2D(pool_size=(2,2)))

# Το μοντέλο θα εκπαιδεύσει 93,322 παραμέτρους
# Προσθέτουμε 2ο συνελικτικό στρώμα με 64 φιλτρα διαστάσεων 3X3 με ίδια συνάρτηση ενεργοποίησης
# δεν χρειάζεται ξανά δήλωση για το σχήμα των εισόδων(input_shape)
model.add(layers.Conv2D(64, (3,3), activation = "relu"))
# Ακολουθεί ένα ακόμα στρώμα συγκέντρωσης 2X2
model.add(layers.MaxPooling2D((2, 2)))

# τέλος, φτιάχνουμε ένα 3ο συνελικτικό στρώμα με 64 φίλτρα διαστάσεων 3X3
model.add(layers.Conv2D(64, (3,3), activation="relu"))
```

```
# Αφού φτιάξαμε τη βάση μας, για να περάσουν οι είσοδοι στο επόμενο στρώμα,
# πρέπει η έξοδος να γίνει "επίπεδη", δηλαδή δύο διαστάσεων
model.add(layers.Flatten())
# Δηλώνουμε τον αριθμό (64) πλήρως συνδεδεμένων νευρώνων, όσοι και τα φίλτρα μας
model.add(layers.Dense(64, activation = "relu"))
# το τελευταίο στρώμα, είναι στρώμα ταξινόμησης με 10 νευρώνες
# όσες και οι κλάσεις των δεδομένων μας "αριθμοί από 0 ως 9"
# κάθε νευρώνας από τους 10, υπολογίζει την πιθανότητα κάθε εικόνας
# να ανήκει σε κάποια από τις κλάσεις
model.add(layers.Dense(10, activation = "softmax"))
# επισκόπιση του μοντέλου
model.summary()
```

```
Our data contain 60000 images for training and 10000 images for testing

(60000, 28, 28)
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_33 (Conv2D)          (None, 26, 26, 32)        320

 max_pooling2d_22 (MaxPoolin  (None, 13, 13, 32)       0
 g2D)

 conv2d_34 (Conv2D)          (None, 11, 11, 64)        18496

 max_pooling2d_23 (MaxPoolin  (None, 5, 5, 64)         0
 g2D)

 conv2d_35 (Conv2D)          (None, 3, 3, 64)          36928

 flatten_11 (Flatten)        (None, 576)               0

 dense_22 (Dense)            (None, 64)                36928

 dense_23 (Dense)            (None, 10)                650

=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
```

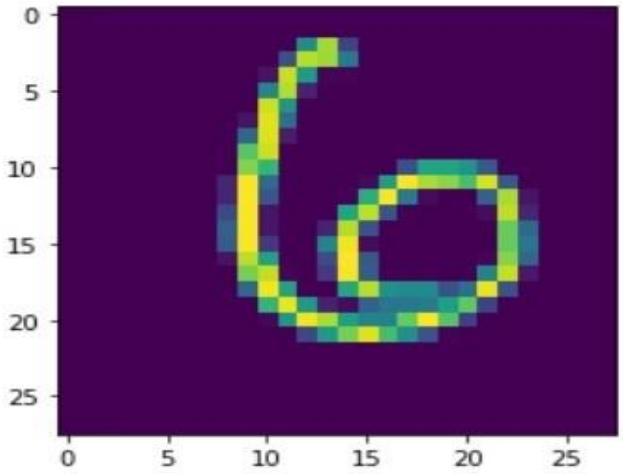
```
# Εκπαίδευση μοντέλου

# κατά τη μεταγλώττιση εδώ, δηλώσαμε τη μέθοδο βελτιστοποίησης,

# τη συνάρτηση απώλειας και

# τη μέτρηση για την ακρίβεια του μοντέλου μας

model.compile(optimizer="adam", loss= "sparse_categorical_crossentropy", metrics=
(["accuracy"]))

# Εκπαιδεύουμε το μοντέλο, χωρίζοντας τις εισόδους σε batches των 100

# Αυτό συμβάλλει στην ταχύτερη εκπαίδευση του μοντέλου.

history=model.fit(train_images, train_labels, epochs=10, batch_size=100, validation_split=0.14)
```

```
Epoch 1/10
516/516 [==============================] - 41s 79ms/step - loss: 0.2439 - accuracy: 0.9253 - val_loss: 0.0750 - val_accuracy: 0.9788
Epoch 2/10
516/516 [==============================] - 40s 78ms/step - loss: 0.0598 - accuracy: 0.9816 - val_loss: 0.0636 - val_accuracy: 0.9818
Epoch 3/10
516/516 [==============================] - 41s 80ms/step - loss: 0.0409 - accuracy: 0.9868 - val_loss: 0.0511 - val_accuracy: 0.9852
Epoch 4/10
516/516 [==============================] - 42s 81ms/step - loss: 0.0311 - accuracy: 0.9900 - val_loss: 0.0435 - val_accuracy: 0.9870
Epoch 5/10
516/516 [==============================] - 41s 79ms/step - loss: 0.0261 - accuracy: 0.9917 - val_loss: 0.0378 - val_accuracy: 0.9888
Epoch 6/10
516/516 [==============================] - 41s 79ms/step - loss: 0.0203 - accuracy: 0.9933 - val_loss: 0.0380 - val_accuracy: 0.9899
Epoch 7/10
516/516 [==============================] - 42s 81ms/step - loss: 0.0184 - accuracy: 0.9942 - val_loss: 0.0363 - val_accuracy: 0.9890
Epoch 8/10
516/516 [==============================] - 41s 79ms/step - loss: 0.0152 - accuracy: 0.9948 - val_loss: 0.0454 - val_accuracy: 0.9882
Epoch 9/10
516/516 [==============================] - 41s 79ms/step - loss: 0.0140 - accuracy: 0.9953 - val_loss: 0.0407 - val_accuracy: 0.9890
Epoch 10/10
516/516 [==============================] - 41s 79ms/step - loss: 0.0111 - accuracy: 0.9965 - val_loss: 0.0398 - val_accuracy: 0.9889
313/313 [==============================] - 3s 10ms/step - loss: 0.0338 - accuracy: 0.9896
0.033750660717487335  -  0.9896000027656555
313/313 [==============================] - 3s 10ms/step - loss: 0.0338 - accuracy: 0.9896


 accuracy :  0.9896
```

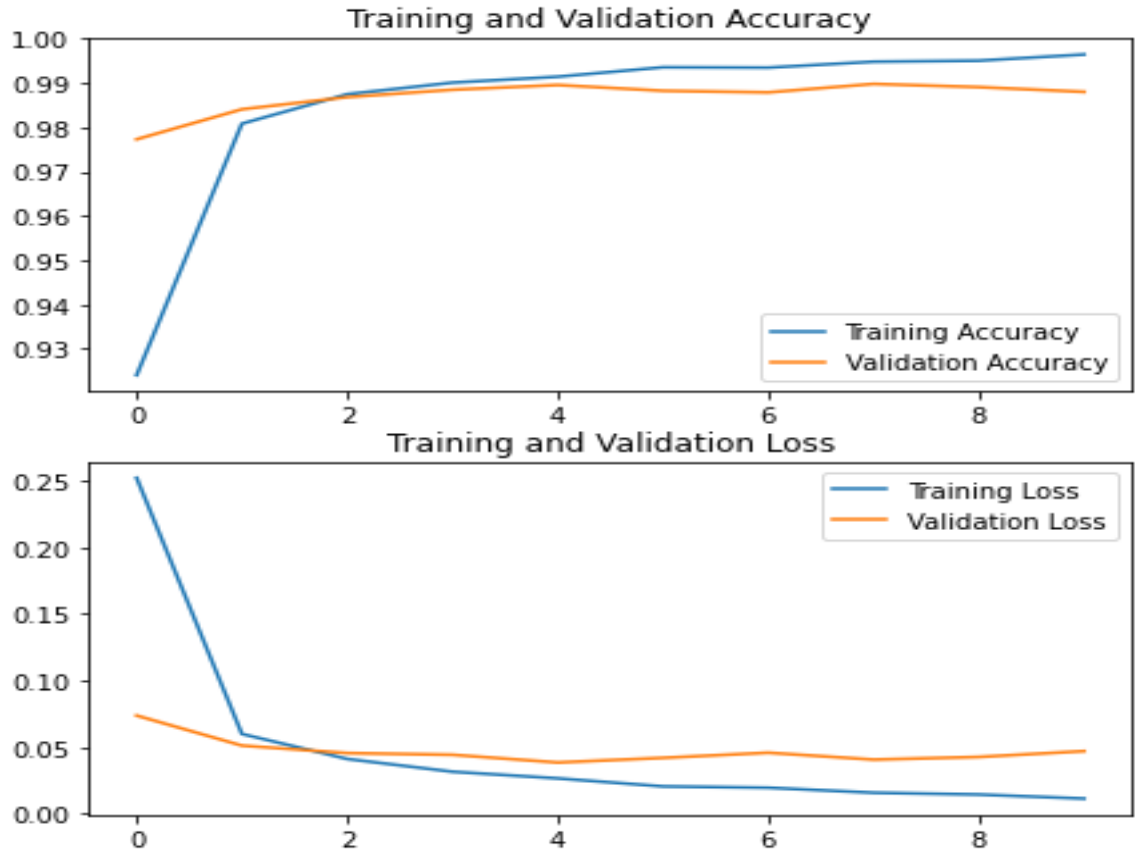
```
# Αξιολόγηση του μοντέλου, ως προς την ακρίβεια πρόβλεψης και τις αποκλίσεις
#  Μετά την εκπαίδευση κάνουμε τυχαίο έλεγχο (στη θέση 100) για την ακρίβεια πρόβλεψης

# Εμφανίζει την ακρίβεια που έχει επιτύχει το μοντέλο μας
```

```
model.evaluate(test_images, test_labels)
y_pred= model.predict(test_images)
y_pred = np.argmax(y_pred, axis=1).astype(int)
from sklearn.metrics import accuracy_score
acc_score= accuracy_score(y_pred,test_labels)
print("\n")
print ( " accuracy : ", acc_score)
print("\n")
print(" prediction test for image number: ",y_pred[100],"  label : ",test_labels[100])
plt.figure()
plt.imshow(test_image[100])
print(" If the image shows ", test_labels[100]," then ok, else false ")
print("\n")
```

```
prediction test for image number:  6   label :  6
If the image shows  6  then ok, else false
```

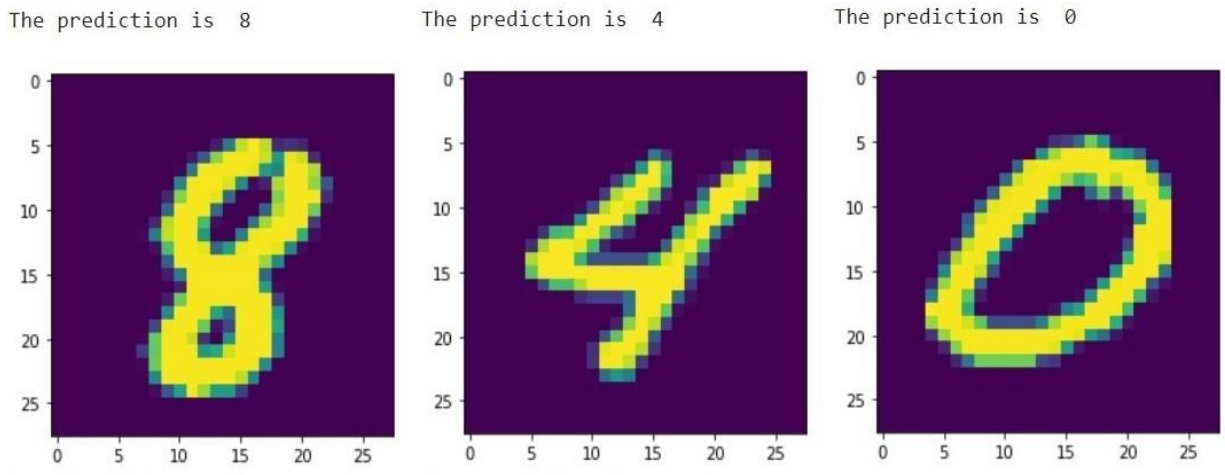


```
# Καμπύλη μάθησης - Learning Curve

epochs=10
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs_range = range(epochs)
plt.figure(figsize=(7, 7))

plt.subplot(2, 1, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation  Loss')

plt.show()
```




```
# Σώζουμε το εκπαιδευμένο μοντέλο μας.

model.save("Number's_image_classification.h5")

# h.5 είναι η επέκταση αρχείου που χρησιμοποιείται από προεπιλογή στο Tensorflow

# Δοκιμάζουμε την προβλεπτική

# ικανότητα του μοντέλου σε τυχαία δεδομένα.

import numpy as np
from keras.preprocessing import image
import matplotlib.pyplot as plt
%matplotlib inline
import random
for i in range(3):
    num_test= tf.random.uniform(
    shape=(), minval=0, maxval=10000, dtype=tf.int32, seed=None, name=None)
```

```
test_image = np.expand_dims(test_images[num_test], axis=0)
result = model.predict(test_image)
plt.figure(clear=True)
print("The prediction is ", result.argmax(), "  "+"\n")
plt.imshow(test_image.reshape(28,28))
plt.show()
```



LSTM model

```
# Εισάγουμε τις βιβλιοθήκες

import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# Βήμα 2ο : Προεργασία στα δεδομένα

# Εισαγωγή δεδομένων από mnist

from keras.datasets import mnist
(x_train,y_train),(x_test,y_test)= mnist.load_data()
```

```
# data από τη διεύθυνση: https://storage.Googleapis.com/tensorflow/tf-keras-datasets/mnist.npz

# Διαχωρισμός δεδομένων σε σύνολα εκπαίδευσης και ελέγχου

x_train.shape, x_test.shape

# output : ((60000, 28, 28), (10000, 28, 28))

# x_test και x_train περιέχουν όλες τις εικόνες(διαστάσεων 28X28) των αριθμών

# y_train and y_test περιέχουν τις ετικέτες των αριθμών

x_train.min(), x_train.max()

y_train.min(),y_train.max()

# Κανονικοποιούμε τα δεδομένα μας

x_train=x_train/255.0
x_test=x_test/255.0

# Βήμα 3ο : Χτίζουμε LSTM

# Καθορισμός μοντέλου RNN (Sequential)

model=tf.keras.models.Sequential()

# Προσθέτουμε το 1ο layer LSTM

model.add(tf.keras.layers.LSTM(units=128,activation='relu',return_sequences=True,input_shape=(28,28)))

# Προσθέτουμε Dropout layer

model.add(tf.keras.layers.Dropout(0.2))

# Η χρήση Dropout layer βοηθά στην πρόληψη εμφάνισης υπερπροσαρμογής

# Η προσθήκη ενός Dense layer αμέσως μετά το 1ο LSTM layer,

# δημιουργεί προβλήματα ανάγνωσης των επιστροφών (return_sequences)

# γι' αυτό προσθέτουμε
```

```
# 2ο LSTM layer
model.add(tf.keras.layers.LSTM(units=128,activation='relu'))
# Αφαιρούμε την παράμετρο return_sequences, διότι δεν θα προσθέσω άλλο επίπεδο LSTM
# Επίσης θα προσθέτουμε Dense layer, μετά το Dropout layer
# που χρησιμοποιείται για την αποφυγή υπερπροσαρμογής των παραμέτρων μας
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(units=32,activation='relu'))
model.add(tf.keras.layers.Dropout(0.2))
# Προσθέτουμε το output layer, με συνάρτηση ενεργοποίησης 'softmax'
model.add(tf.keras.layers.Dense(units=10,activation='softmax'))
# Για δυαδικές (binary) εξόδους, χρησιμοποιούμε σιγμοειδής (sigmoid)
# συνάρτηση ενεργοποίησης
# Για περισσότερες εξόδους χρησιμοποιούμε τη συνάρτηση ενεργοποίησης softmax
# Επισκόπιση μοντέλου
model.summary()
```

```
2.8.0
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 28, 128)           80384

 dropout (Dropout)           (None, 28, 128)           0

 lstm_1 (LSTM)               (None, 128)               131584

 dropout_1 (Dropout)         (None, 128)               0

 dense (Dense)               (None, 32)                4128

 dropout_2 (Dropout)         (None, 32)                0

 dense_1 (Dense)             (None, 10)                330

=================================================================
Total params: 216,426
Trainable params: 216,426
Non-trainable params: 0
```

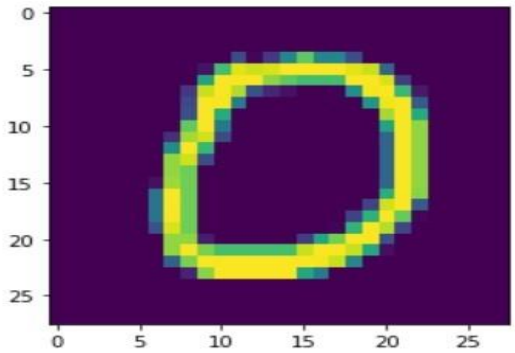
```
# Το μοντέλο θα εκπαιδεύσει 216,426 παραμέτρους

# Δηλώνουμε βελτιστοποιητή Adam, με βήμα εκμάθησης 0,001

opt = tf.keras.optimizers.Adam(learning_rate=0.001)

# Μεταγλώττιση μοντέλου

model.compile(optimizer=opt, loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Η παράμετρος loss θα οδηγήσει τον βελτιστοποιητή στο ολικό ελάχιστο

# και το meterics θα υπολογίσει το ποσοστό ακρίβειας του μοντέλου

# Βήμα 4ο : Εκπαίδευση μοντέλου

history =  model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test), batch_size=100,
validation_split=0.14)
```

```
Epoch 1/10
1875/1875 [==============================] - 113s 59ms/step - loss: 0.5773 - accuracy: 0.8180 - val_loss: 0.1192 - val_accuracy: 0.9666
Epoch 2/10
1875/1875 [==============================] - 111s 59ms/step - loss: 0.1566 - accuracy: 0.9582 - val_loss: 0.0864 - val_accuracy: 0.9728
Epoch 3/10
1875/1875 [==============================] - 112s 60ms/step - loss: 0.1073 - accuracy: 0.9711 - val_loss: 0.0803 - val_accuracy: 0.9781
Epoch 4/10
1875/1875 [==============================] - 113s 60ms/step - loss: 0.0871 - accuracy: 0.9770 - val_loss: 0.0672 - val_accuracy: 0.9816
Epoch 5/10
1875/1875 [==============================] - 113s 60ms/step - loss: 0.0746 - accuracy: 0.9804 - val_loss: 0.0574 - val_accuracy: 0.9838
Epoch 6/10
1875/1875 [==============================] - 112s 60ms/step - loss: 0.0591 - accuracy: 0.9839 - val_loss: 0.0635 - val_accuracy: 0.9840
Epoch 7/10
1875/1875 [==============================] - 110s 59ms/step - loss: 0.0539 - accuracy: 0.9861 - val_loss: 0.0590 - val_accuracy: 0.9852
Epoch 8/10
1875/1875 [==============================] - 111s 59ms/step - loss: 0.0492 - accuracy: 0.9869 - val_loss: 0.0452 - val_accuracy: 0.9860
Epoch 9/10
1875/1875 [==============================] - 110s 59ms/step - loss: 0.0427 - accuracy: 0.9890 - val_loss: 0.0480 - val_accuracy: 0.9880
Epoch 10/10
1875/1875 [==============================] - 109s 58ms/step - loss: 0.0399 - accuracy: 0.9894 - val_loss: 0.0382 - val_accuracy: 0.9893
0 0
```

```
#  60000 δείγματα εκπαίδευσης και 10000 επικύρωσης του μοντέλου, σε 10 επαναλήψεις

#  Μετά την εκπαίδευση κάνουμε τυχαίο έλεγχο για την ακρίβεια πρόβλεψης

y_pred= model.predict(x_test)

y_pred = np.argmax(y_pred, axis=1).astype(int)
```

```
from sklearn.metrics import accuracy_score
acc_score= accuracy_score(y_pred,y_test)
print("\n")
print ( " accuracy : ", acc_score)
print("\n")
print(" prediction test for image number: ",y_pred[10],"  label : ",y_test[10])
plt.imshow(x_test[10])
print()
print(" If the image shows ", y_test[10]," then ok, else false ")
print("\n")
```

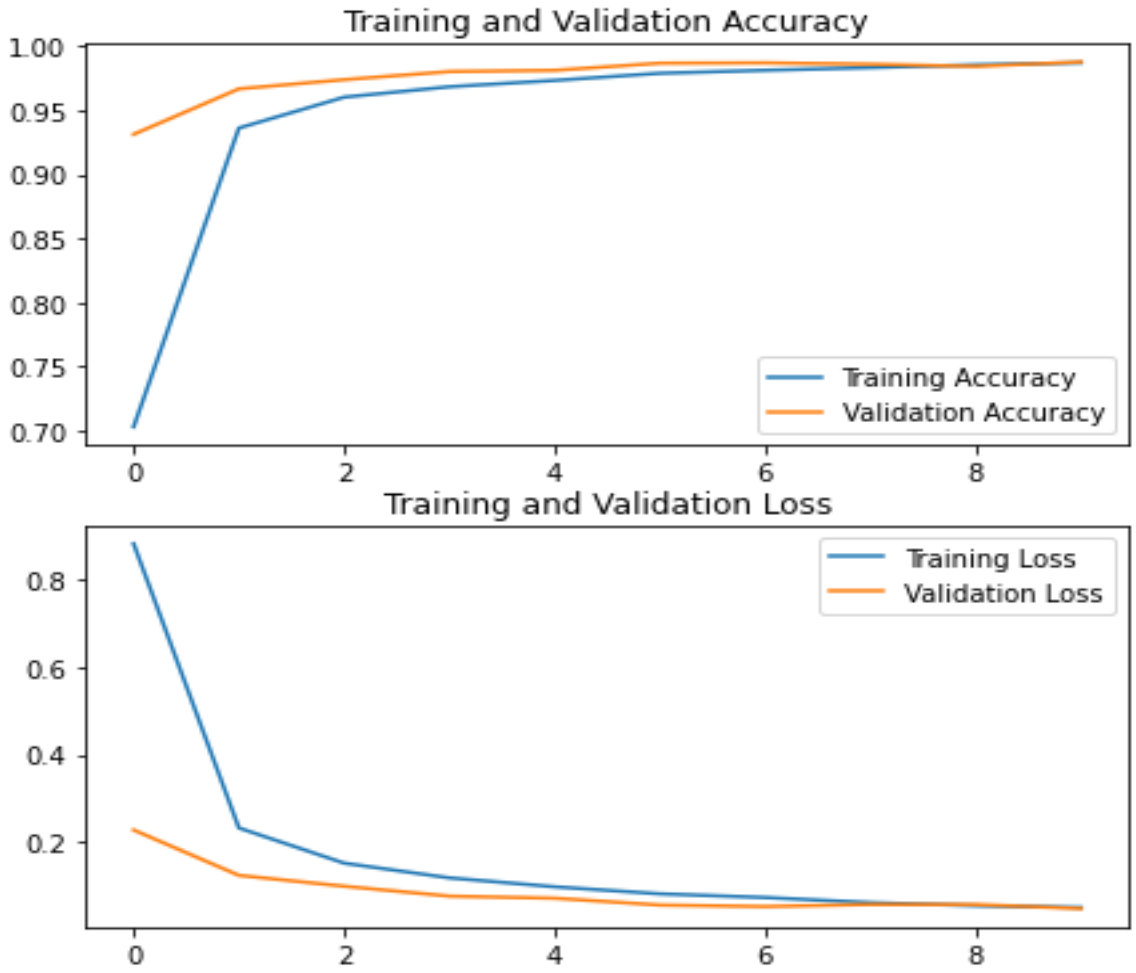
```
prediction test for image number:  0   label :  0
If the image shows  0  then ok, else false
```



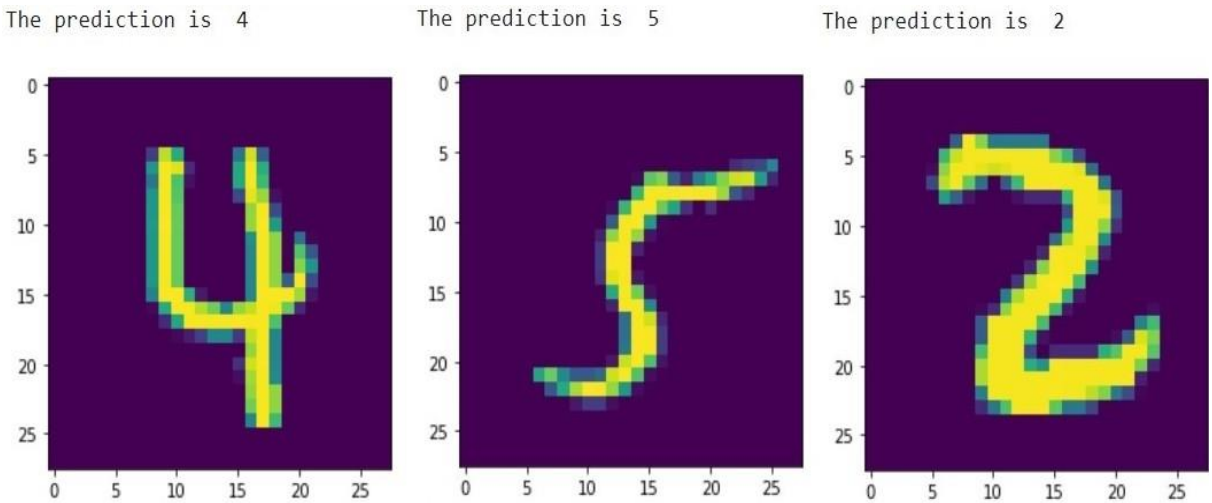
Βήμα 5 Καμπύλη μάθησης - Learning Curve

```
epochs=10
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs_range = range(epochs)
```

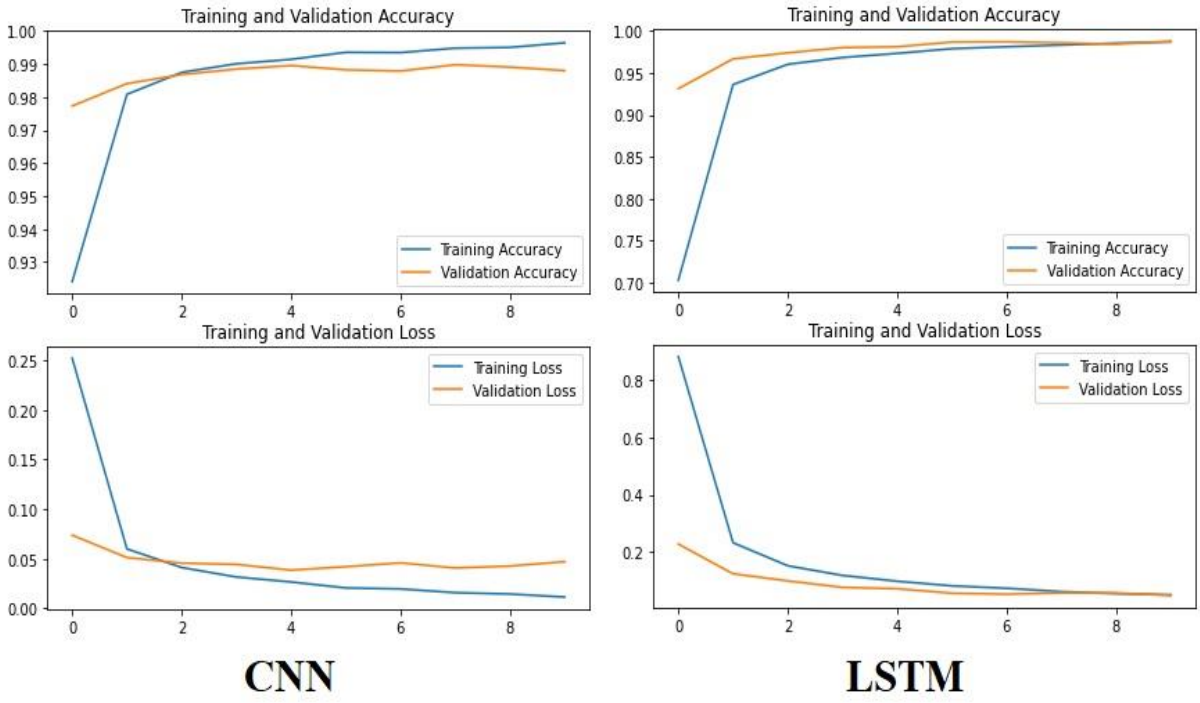
```
plt.figure(figsize=(7, 7))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')
plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



```
# Σώζουμε το εκπαιδευμένο μοντέλο μας

model.save("Number's_image_LSTM_classification.h5")

# h.5 είναι η επέκταση αρχείου που χρησιμοποιείται από προεπιλογή στο Tensorflow

# Δοκιμάζουμε το μοντέλο για την προβλεπτική του ικανότητα σε τυχαίες θέσεις δεδομένων

import numpy as np
from keras.preprocessing import image
import matplotlib.pyplot as plt
%matplotlib inline
import random

for i in range(3):
    num_test= tf.random.uniform(
    shape=(), minval=0, maxval=10000, dtype=tf.int32, seed=None, name=None)
    test_image = np.expand_dims(x_test[num_test], axis=0)
    result = model.predict(test_image)
    plt.figure(clear=True)
    print("The prediction is ", result.argmax(), "  "+"\n")
    plt.imshow(test_image.reshape(28,28))
    plt.show()
```



3.3 Παρατηρήσεις για τα αποτελέσματα των μοντέλων



Στην 1η φάση εκπαίδευσης το CNN μοντέλο επιτυγχάνει ακρίβεια πάνω από 98%, ενώ το LSTM γύρω στο 94%. Κατά την αξιολόγηση, το μοντέλο LSTM δείχνει πιο βελτιωμένο σε σχέση με το CNN, χωρίς όμως να το ξεπερνάει σε ακρίβεια. Στις επόμενες φάσεις εκπαίδευσης το μοντέλο CNN εμφανίζει στις τιμές ακρίβειας (accuracy), αρνητική απόκλιση μεταξύ αξιολόγησης και εκπαίδευσης. Σε μεγαλύτερες αποκλίσεις, εξετάζουμε το ενδεχόμενο υπερπροσαρμογής (overfitting) των παραμέτρων μας. Οι τιμές στην αξιολόγηση του μοντέλου LSTM σχετικά με την ακρίβεια, συγκλίνουν σε αυτές της εκπαίδευσης σε αντίθεση με το CNN μοντέλο. Η ακρίβεια πρόβλεψης στο μοντέλο CNN είναι 98.9% και στο μοντέλο LSTM 98,8%. Δεν φαίνονται σημαντικές οι διαφορές αυτές για το μέγεθος των δεδομένων με τα οποία τροφοδοτήσαμε τα μοντέλα μας. Όσο όμως αυξάνεται η ποσότητα των δεδομένων, τόσο μεγαλύτερη αξία αποκτούν τέτοιες διαφορές. Σημαντική διαφορά εντοπίζουμε στον χρόνο εκπαίδευσης. Το μοντέλο CNN εκπαιδεύεται στον μισό περίπου χρόνο (41s έναντι 83s)σε σχέση με το LSTM. Ο βασικός λόγος είναι οι υπερδιπλάσιες παραμέτρους που δημιουργεί το μοντέλο LSTM (216.426 παραμέτρους έναντι 93.322) σε σχέση με το μοντέλο CNN.

3.4 Επεξεργασία φυσικής γλώσσας (NLP natural language processing)

Η επεξεργασία φυσικής γλώσσας αφορά τη χρήση μηχανών για τον χειρισμό της φυσικής γλώσσας. Η μελέτη της επεξεργασίας φυσικής γλώσσας υπάρχει εδώ και περισσότερα από 50 χρόνια. Το περίφημο τεστ Turing για τη γενική τεχνητή νοημοσύνη χρησιμοποιεί αυτή τη γλώσσα.

Σύμφωνα με την IBM, 2,5 exabyte (1 exabyte = 1.000.000.000 gigabyte) δεδομένων παράγονταν κάθε μέρα το 2017 και αυτό αυξάνεται κάθε μέρα. Από όλα αυτά τα δεδομένα, ένα μεγάλο μέρος είναι αδόμητο κείμενο και ομιλία, καθώς δημιουργούνται εκατομμύρια email και περιεχόμενο μέσων κοινωνικής δικτύωσης και πραγματοποιούνται τηλεφωνικές κλήσεις καθημερινά. Ο στόχος του NLP είναι να κάνει τις μηχανές να κατανοούν την προφορική και γραπτή γλώσσα μας. Το NLP είναι πανταχού παρόν και αποτελεί ήδη ένα μεγάλο μέρος της ανθρώπινης ζωής. Εικονικοί Βοηθοί (VAs), όπως το Google Assistant, η Cortana και το Apple Siri, είναι σε μεγάλο βαθμό συστήματα NLP. Πολλές εργασίες NLP πραγματοποιούνται όταν κάποιος ρωτά έναν VA: «Μπορείς να μου δείξεις ένα καλό ιταλικό εστιατόριο κοντά;». Πρώτον, το VA πρέπει να μετατρέψει την έκφραση σε κείμενο (δηλαδή ομιλία σε κείμενο). Στη συνέχεια, πρέπει να κατανοήσει τη σημασιολογία του αιτήματος (για παράδειγμα, ο χρήστης αναζητά ένα καλό εστιατόριο με ιταλική κουζίνα) και να διατυπώσει ένα δομημένο αίτημα (για παράδειγμα, κουζίνα = ιταλική, βαθμολογία = 3-5, απόσταση < 10 km). Το VA πρέπει να αναζητήσει εστιατόρια με φιλτράρισμα με βάση την τοποθεσία και την κουζίνα και στη συνέχεια, να ταξινομήσει τα εστιατόρια με βάση τις αξιολογήσεις που έλαβε. Για τον υπολογισμό μίας συνολικής βαθμολογίας για ένα εστιατόριο, ένα καλό σύστημα NLP μπορεί να εξετάσει τόσο τη βαθμολογία όσο και την περιγραφή κειμένου που παρέχονται από κάθε χρήστη. Τέλος, μόλις ο χρήστης βρεθεί στο εστιατόριο, το VA μπορεί να βοηθήσει τον χρήστη μεταφράζοντας διάφορα στοιχεία μενού από τα ιταλικά στα αγγλικά. Αυτό το παράδειγμα δείχνει ότι το NLP έχει γίνει αναπόσπαστο μέρος της ανθρώπινης ζωής. Θα έπρεπε να γίνει κατανοητό ότι το NLP είναι ένα εξαιρετικά απαιτητικό πεδίο έρευνας. Κάθε γλώσσα έχει τη δική της γραμματική, σύνταξη και λεξιλόγιο. Ως εκ τούτου, η επεξεργασία δεδομένων κειμένου περιλαμβάνει διάφορες σύνθετες εργασίες, όπως ανάλυση κειμένου (για παράδειγμα, αποτύπωση και δημιουργία βασικών στοιχείων), μορφολογική ανάλυση, αποσαφήνιση της έννοιας της λέξης και κατανόηση της γραμματικής δομής μίας γλώσσας.

3.4.1 Εργασίες επεξεργασίας Φυσικής Γλώσσας

Το NLP έχει πληθώρα εφαρμογών στον πραγματικό κόσμο. Ένα καλό σύστημα NLP είναι αυτό που εκτελεί πολλές εργασίες NLP. Όταν ψάχνουμε για τον καιρό σήμερα στο Google ή χρησιμοποιούμε τη μετάφραση *Google* για να μάθουμε πώς να πούμε "Πώς είσαι;" στα γαλλικά, βασίζεστε σε ένα υποσύνολο τέτοιων εργασιών στο NLP. Όταν έχουμε ένα σώμα κειμένου που είναι σε ακατάλληλη μορφή (αδόμητο), θα εκτελέσουμε όλα τα βήματα της προεπεξεργασίας για να πάρουμε κάποιο νόημα από το κείμενο. Θα απαριθμήσουμε μερικές από τις πιο διαδεδομένες εργασίες, με βάση την αγγλική γλώσσα.

- **Tokenization** : σημαίνει τη διάδοση του κειμένου σε διακριτές, ελάχιστες, με ουσιαστικό νόημα ενότητες. Αυτές οι ενότητες μπορεί μερικές φορές να είναι λέξεις ή προτάσεις. Διαχωρίζουμε ένα σώμα κειμένου σε λέξεις ή προτάσεις. Για παράδειγμα :

```
import nltk

from nltk import word_tokenize
```

Η word_tokenize() μέθοδος χρησιμοποιείται για να χωρίσει την πρόταση σε λέξεις/δείκτες. Πρέπει να προσθέσουμε μία πρόταση ως είσοδο στη word_tokenize() μέθοδο, ώστε να εκτελέσει τη δουλειά της. Το αποτέλεσμα θα ήταν μία λίστα, την οποία θα αποθηκεύσουμε σε μία word μεταβλητή.

```
words = word_tokenize("I am reading a book")

print(words)
```

Ο κώδικας δημιουργεί την ακόλουθη έξοδο: [’I’, ‘am’, ‘reading’, ‘a’, ‘book’]

- **Ετικέτες Part-of-Speech (PoS)** : Η προσθήκη ετικετών PoS είναι η εργασία της ανάθεσης λέξεων στα αντίστοιχα μέρη του λόγου. Μπορεί να είναι βασικές ετικέτες όπως ουσιαστικό, ρήμα, επίθετο, επίρρημα και πρόθεση. Για παράδειγμα, αν κοιτάξουμε την πρόταση "Ο ουρανός είναι μπλε", θα λάβουμε τέσσερις δείκτες "Ο", "ουρανός", "είναι" και "μπλε" , με τη βοήθεια του tokenization. Χρησιμοποιώντας το **PoS tagger** , προσθέτουμε ετικέτες σε μέρη της ομιλίας σε κάθε λέξη. Αυτό θα φαίνεται ως εξής:

[('Ο', 'DT'), ('ουρανός', 'NN'), ('είναι', 'VBZ'), ('μπλε', 'JJ')]

DT = προσδιοριστής

NN = ουσιαστικό, κοινό, ενικό ή μάζα

VBZ = ρήμα, ενεστώτα, 3ο ενικό πρόσωπο

JJ = Επίθετο

```
import nltk
from nltk import word_tokenize
```

Για την εύρεση των σημείων στην πρόταση, χρησιμοποιούμε τη word_tokenize() μέθοδο.

```
words = word_tokenize("I am reading NLP Fundamentals")
print(words)
```

Ο κώδικας δημιουργεί την ακόλουθη έξοδο: [‘I’, ‘am’, ‘reading’, ‘NLP’, ‘fundamentals’]

Για να βρούμε το PoS για κάθε λέξη, χρησιμοποιούμε τη **pos_tag()** μέθοδο της **nltk** βιβλιοθήκης.

```
nltk.pos_tag(words)
```

Ο κώδικας δημιουργεί την ακόλουθη έξοδο:

[(‘I’, ‘PRP’), (‘am’, ‘VBP’), (‘reading’, ‘VBG’), (‘NLP’, ‘NNP’), ‘fundamentals’, ‘NNS’)]

Στην προηγούμενη έξοδο, μπορούμε να δούμε ότι για κάθε διακριτικό, έχει εκχωρηθεί ένα PoS. Το PRP σημαίνει προσωπική αντωνυμία , το VBP σημαίνει ρήμα παρόν , το VGB σημαίνει ρήμα γερουνδίου , το NNP σημαίνει το σωστό ουσιαστικό ενικού και το NNS σημαίνει ουσιαστικό πληθυντικού .

- **Διακοπή κατάργησης λέξης** (Stop Word Removal) : Οι λέξεις διακοπής είναι κοινές λέξεις που χρησιμοποιούνται απλώς για να υποστηρίξουν την κατασκευή προτάσεων. Καταργούμε τις λέξεις τερματισμού από την ανάλυσή μας, καθώς δεν επηρεάζουν το νόημα των προτάσεων στις οποίες υπάρχουν. Παραδείγματα λέξεων τερματισμού

περιλαμβάνουν a, am και the. Δεδομένου ότι εμφανίζονται πολύ συχνά και η παρουσία τους δεν έχει μεγάλη επίδραση στην έννοια της πρότασης, πρέπει να αφαιρεθούν. Παράδειγμα αφαίρεσης λέξεων:

```
import nltk
nltk.download('stopwords')
from nltk import word_tokenize
from nltk.corpus import stopwords
```

Για να ελέγξουμε τη λίστα των λέξεων λήξης που παρέχονται για τη αγγλική γλώσσα, τη μεταβιβάζουμε ως παράμετρο στη words()

```
stop_words = stopwords.words('English')
```

Στον κώδικα, η λίστα των λέξεων λήξης που παρέχονται από τη αγγλική γλώσσα αποθηκεύεται στη stop_words μεταβλητή.

```
print(stop_words)
```

Ο κώδικας δημιουργεί την ακόλουθη έξοδο:

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'y
ourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those',
'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'a
n', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'b
etween', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
f', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'ar
en', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "have
n't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

Για να αφαιρέσουμε τις λέξεις διακοπής από μία πρόταση, εκχωρούμε πρώτα μία συμβολοσειρά στη sentence μεταβλητή και την ονομάζουμε σε λέξεις χρησιμοποιώντας τη word_tokenize() μέθοδο.

```
sentence = "I am learning Python. It is one of the most popular programming languages"
sentence_words = word_tokenize(sentence)
```

```
print(sentence_words)
```

Ο κώδικας δημιουργεί την ακόλουθη έξοδο:

['I', 'am', 'learning', 'Python', '.', 'It', 'is', 'one', 'of', 'the', 'most', 'popular', 'programming', 'languages']

Για να αφαιρέσουμε τις ενδιάμεσες λέξεις, πρώτα πρέπει να περάσουμε σε βρόχο κάθε λέξη της πρότασης, να ελέγξουμε αν υπάρχουν τερματικές λέξεις και τέλος τις συνδυάζουμε για να σχηματίσουμε μία πλήρη πρόταση.

```
sentence_no_stops = ' '.join([word for word in sentence_words if word not in stop_words])
```

Για να ελέγξουμε αν οι λέξεις stop, έχουν φιλτραριστεί από την πρόταση μας, εκτυπώνουμε τη sentence_no_stops μεταβλητή.

```
print(sentence_no_stops)
```

Ο κώδικας δημιουργεί την ακόλουθη έξοδο:

I learning Python . It one popular programming languages

Λέξεις τερματισμού όπως " am ", " is ", " of", " the ", " most " φιλτράρονται και κείμενο χωρίς λέξεις τερματισμού παράγεται ως έξοδο.

- **Κανονοποίηση κειμένου** (Text Normalization): Υπάρχουν ορισμένες λέξεις που γράφονται, προφέρονται και αναπαρίστανται διαφορετικά, για παράδειγμα λέξεις όπως Moumbai και Bombay ή ΗΠΑ και Ηνωμένες Πολιτείες Αμερικής. Αν και είναι διαφορετικά, σημαίνουν το ίδιο πράγμα. Υπάρχουν επίσης λέξεις διαφορετικών μορφών που πρέπει να μετατραπούν σε βασικές μορφές. Για παράδειγμα, λέξεις όπως "does" και "doing", όταν μετατρέπονται στη βασική τους μορφή, γίνονται "does". Σε αυτές τις γραμμές, η κανονικοποίηση κειμένου είναι μία διαδικασία κατά την οποία διαφορετικές παραλλαγές κειμένου μετατρέπονται σε μία τυπική μορφή. Πρέπει να κάνουμε κανονικοποίηση, καθώς υπάρχουν κάποιες λέξεις που μπορεί να σημαίνουν το ίδιο πράγμα η μία με την άλλη. Υπάρχουν διάφοροι τρόποι κανονικοποίησης του κειμένου, όπως η ορθογραφία και η λημματοποίηση.

Θα προσπαθήσουμε να αντικαταστήσουμε επιλεγμένες λέξεις με νέες λέξεις, χρησιμοποιώντας τη replace() συνάρτηση και τελικά να παράγουμε το κανονικοποιημένο κείμενο.

```
sentence = "I visited US from UK on 22-10-18"
```

Θέλουμε να αντικαταστήσουμε το " US" με " United States", το " UK" με " United Kingdom" και το " 18" με " 2018". Για να το κάνουμε αυτό, χρησιμοποιούμε τη replace() συνάρτηση και αποθηκεύουμε την ενημερωμένη έξοδο στη normalized_sentence μεταβλητή.

```
normalized_sentence = sentence.replace("US", "United States").replace("UK", "United Kingdom").replace("-18", "-2018")

print(normalized_sentence)
```

Η έξοδος που παράγεται είναι η ακόλουθη:

"I visited United States from United Kingdom on 22-10-2018"

- **Ορθογραφία** (Spelling Correction): Η ορθογραφία είναι μία από τις πιο σημαντικές εργασίες σε κάθε έργο NLP. Μπορεί να είναι χρονοβόρο, αλλά χωρίς αυτό υπάρχουν μεγάλες πιθανότητες να χάσουμε τις απαιτούμενες πληροφορίες. Χρησιμοποιούμε τη βιβλιοθήκη Python "autocorrect" για να διορθώσουμε τα ορθογραφικά λάθη.

Εκτελούμε ορθογραφική διόρθωση σε μία λέξη και μία πρόταση, με τη βοήθεια της autocorrect βιβλιοθήκης " της Python.

```
import nltk
from nltk import word_tokenize
from autocorrect import spell
```

Για να διορθώσουμε την ορθογραφία μίας λέξης, περνάμε μία λανθασμένη λέξη ως παράμετρο στη spell() συνάρτηση.

```
spell('Natureal')
```

Ο κώδικας δημιουργεί την ακόλουθη έξοδο: ‘Natural’

Για να διορθώσουμε την ορθογραφία μίας πρότασης, πρέπει πρώτα να τη χωρίσουμε σε λέξεις. Μετά από αυτό, κάνουμε βρόχο σε κάθε λέξη στο sentence, τις διορθώνουμε αυτόματα και τέλος τις συνδυάζουμε.

```
sentence = word_tokenize("Ntural Luanguage Processin deals with the art of extracting insightes from Natural Languaes")
```

Τώρα που έχουμε τον διαχωρισμό των λέξεων, κάνουμε βρόχο μέσα από κάθε διακριτικό στο sentence, τις διορθώνουμε και τις εκχωρούμε σε νέα μεταβλητή.

```
sentence_corrected = ' '.join([spell(word) for word in sentence])
```

```
print(sentence_corrected)
```

Ο κώδικας δημιουργεί την ακόλουθη έξοδο:

“Natural Language Procession deals with the art of extracting insights from Natural Languages”

- **Στέλεχο**ς (Stemming): Σε γλώσσες όπως τα αγγλικά, οι λέξεις μετατρέπονται σε διάφορες μορφές όταν χρησιμοποιούνται σε μία πρόταση. Για παράδειγμα, η λέξη "product" μπορεί να μετατραπεί σε "production" όταν αναφέρεται στη διαδικασία κατασκευής ή να μετατραπεί σε "products" σε μορφή πληθυντικού. Είναι απαραίτητο να μετατραπούν αυτές οι λέξεις στη βασική τους μορφή, καθώς έχουν την ίδια σημασία. Το stemming είναι μία διαδικασία που μας βοηθά να το κάνουμε.

```
import nltk
stemmer = nltk.stem.PorterStemmer()
```

Τώρα θέτουμε τις παρακάτω λέξεις ως παραμέτρους στη stem() μέθοδο.

```
stemmer.stem("production")
```

Όταν η είσοδος είναι " production", δημιουργείται η έξοδος: ‘product’

```
stemmer.stem("coming")
```

Όταν η είσοδος είναι " coming", δημιουργείται η έξοδος: ‘come’

```
stemmer.stem("firing")
```

Όταν η είσοδος είναι " firing", δημιουργείται η έξοδος: ‘fire’

- **Λημματοποίηση** (Lemmatization): Μερικές φορές, η διαδικασία του στελέχους δεν οδηγεί σε κατάλληλα αποτελέσματα. Για να ξεπεράσουμε αυτά τα προβλήματα, χρησιμοποιούμε τη λημματοποίηση για να δημιουργήσουμε τη σωστή μορφή μίας δεδομένης λέξης. Σε αυτή τη διαδικασία, γίνεται ένας πρόσθετος έλεγχος, κοιτάζοντας μέσα από το λεξικό για να εξάγουμε τη βασική μορφή μίας λέξης. Αυτός ο πρόσθετος έλεγχος επιβραδύνει τη διαδικασία, όπως φαίνεται στο παράδειγμα:

```
import nltk
nltk.download('wordnet')
from nltk.stem.wordnet import WordNetLemmatizer
```

Δημιουργία αντικειμένου της WordNetLemmatizer κλάσης.

```
lemmatizer = WordNetLemmatizer()
```

Φέρνουμε τη λέξη στη σωστή της μορφή χρησιμοποιώντας τη lemmatize() μέθοδο.

```
lemmatizer.lemmatize('products')
```

Με την είσοδο " products", δημιουργείται η έξοδος: ‘product’

```
lemmatizer.lemmatize('production')
```

Με την είσοδο " production", δημιουργείται η έξοδος: ‘production’

```
lemmatizer.lemmatize('coming')
```

Με την είσοδο " coming", δημιουργείται η ακόλουθη έξοδος: ‘coming’

- **Αναγνώριση επωνυμίας οντότητας (NER)** : Το NER επιχειρεί να εξαγάγει οντότητες (για παράδειγμα, άτομο, τοποθεσία και οργανισμός) από ένα δεδομένο σώμα κειμένου ή ένα σώμα κειμένου. Οι επώνυμες οντότητες συνήθως δεν υπάρχουν στα λεξικά, έτσι τις αντιμετωπίζουμε ξεχωριστά. Ο κύριος στόχος αυτής της διαδικασίας είναι να αναγνωρίσει τις ονομασμένες οντότητες και να τις αντιστοιχίσει στις κατηγορίες που

έχουν ήδη καθοριστεί. Για παράδειγμα, οι κατηγορίες μπορεί να περιλαμβάνουν ονόματα προσώπων, τοποθεσίες και ούτω καθεξής. Για να κατανοήσουμε καλύτερα αυτή τη διαδικασία, θα δούμε το παράδειγμα:

```
import nltk
from nltk import word_tokenize
nltk.download('maxent_ne_chunker')
nltk.download('words')

sentence = "We are watching a film  directed by Tarantino which is based on life of Draculla."
```

Για να βρούμε τις οντότητες από το προηγούμενο κείμενο,

```
i = nltk.ne_chunk(nltk.pos_tag(word_tokenize(sentence)), binary=True)
[a for a in i if len(a)==1]
```

Ο κώδικας δημιουργεί την ακόλουθη έξοδο:

```
[Tree('NE', [('Tarantino', 'NNP')]), Tree('NE', [('Draculla', 'NNP')])]
```

- **Αποσαφήνιση με έννοια λέξης (WSD)** : Word Sense Disambiguation, είναι το έργο του προσδιορισμού της σωστής σημασίας μίας λέξης. Η σημασία μίας λέξης εξαρτάται από τη συσχέτισή της με άλλες λέξεις της πρότασης. Αυτό σημαίνει ότι δύο ή περισσότερες λέξεις με την ίδια ορθογραφία μπορεί να έχουν διαφορετική σημασία σε διαφορετικά συμφραζόμενα. Αυτό συχνά οδηγεί σε ασάφεια και το παράδειγμα είναι ενδεικτικό του τρόπου με τον οποίο προκαλείται ασάφεια, λόγω της χρήσης της ίδιας λέξης σε διαφορετικές προτάσεις :

"Ξέρει να παίζει κιθάρα", "Ο Γιάννης παίζει μόνο μπάλλα", "Παίζει θέατρο", ποια η έννοια της λέξης "παίζει"; Το WSD είναι κρίσιμο για εργασίες όπως η απάντηση ερωτήσεων.

Εισάγουμε τις απαραίτητες βιβλιοθήκες:

```
import nltk
from nltk.wsd import lesk
from nltk import word_tokenize
```

Δηλώνουμε δύο μεταβλητές sentence1και sentence2, και αντιστοιχίζουμε με τις κατάλληλες συμβολοσειρές:

```
sentence1 = "Ξέρει να παίζει κιθάρα"
sentence2 = "Ο Γιάννης παίζει μόνο μπάλα"
```

Για να βρούμε την έννοια της λέξης «παίζει» στις δύο προηγούμενες προτάσεις, χρησιμοποιούμε τον **lesk** αλγόριθμο που παρέχεται από τη **nltk.wsd** βιβλιοθήκη.

```
print(lesk(word_tokenize(sentence1), 'παίζει'))
print(lesk(word_tokenize(sentence2), 'παίζει'))
```

Ο κώδικας δημιουργεί κάτι σαν την ακόλουθη έξοδο:

```
Synset('μουσική_παίζει.n.03)
Synset('άθλημα_παίζει.n.06)
```

Έτσι, με τη βοήθεια του **lesk** αλγορίθμου, είμαστε σε θέση να αναγνωρίσουμε την έννοια μίας λέξης σε οποιοδήποτε πλαίσιο.

- **Ανίχνευση ορίων προτάσεων (Sentence Boundary Detection).** Η ανίχνευση ορίων πρότασης είναι η μέθοδος ανίχνευσης πού τελειώνει μία πρόταση και πού αρχίζει μία άλλη πρόταση. Πρέπει να πραγματοποιηθούν διάφορες αναλύσεις σε επίπεδο πρότασης, επομένως είναι απαραίτητος ο εντοπισμός των ορίων των προτάσεων. Ακολουθεί ένα παράδειγμα εξαγωγής προτάσεων από μία παράγραφο.

```
import nltk
from nltk.tokenize import sent_tokenize
```

Χρησιμοποιούμε τη **sent_tokenize()** μέθοδο ανίχνευσης προτάσεων σε ένα δεδομένο κείμενο.

```
sent_tokenize("We are reading a book. Do you know who is the writer? It is Bob Knight. Mr Knight lives in Scotland")
```

Ο κώδικας δημιουργεί την ακόλουθη έξοδο:

[‘"We are reading a book.’ ,
‘Do you know who is the writer?’ ,
‘It is Bob Knight.’ ,
‘Mr Knight lives in Scotland"]

3.4.2 Έναρξη ενός έργου NLP

Μπορούμε να χωρίσουμε ένα έργο NLP σε πολλά υποέργα ή φάσεις. Αυτές οι φάσεις ακολουθούνται διαδοχικά, γεγονός που τείνει να αυξάνει τη συνολική αποτελεσματικότητα της διαδικασίας. Ένα έργο NLP πρέπει να περάσει διαδοχικά, από έξι μεγάλες φάσεις:

1. Συλλογή δεδομένων - Data Collection
2. Προεργασία δεδομένων - Data Preprocessing
3. Εξαγωγή χαρακτηριστικών - Feature Extraction
4. Ανάπτυξη μοντέλου - Model development
5. Εκτίμηση Μοντέλου - Model Assessment
6. Εφαρμογή μοντέλου - Model Deployment

Συλλογή δεδομένων. Αυτή είναι η αρχική φάση οποιουδήποτε έργου NLP. Ο μοναδικός μας σκοπός είναι να συλλέγουμε δεδομένα σύμφωνα με τις απαιτήσεις μας. Για αυτό μπορούμε είτε να χρησιμοποιήσουμε υπάρχοντα δεδομένα, είτε να συλλέξουμε δεδομένα από διάφορα διαδικτυακά αποθετήρια ή να δημιουργήσουμε το δικό μας σύνολο δεδομένων ανιχνεύοντας τον ιστό.

Προεπεξεργασία δεδομένων. Μόλις συλλεχθούν τα δεδομένα, πρέπει να τα καθαρίσουμε. Για τη διαδικασία καθαρισμού, χρησιμοποιούμε τα διάφορα βήματα προεπεξεργασίας. Είναι απαραίτητο να καθαρίσουμε τα δεδομένα που συλλέγονται, καθώς τα βρώμικα δεδομένα τείνουν να μειώνουν την αποτελεσματικότητα και την ακρίβεια.

Εξαγωγή χαρακτηριστικών. Οι υπολογιστές κατανοούν μόνο δυαδικά ψηφία: 0 και 1. Έτσι κάθε εντολή που τροφοδοτούμε σε έναν υπολογιστή μετατρέπεται σε δυαδικά ψηφία. Ομοίως, τα μοντέλα μηχανικής μάθησης τείνουν να κατανοούν μόνο αριθμητικά δεδομένα. Καθίσταται απαραίτητο να μετατραπούν τα δεδομένα κειμένου στην ισοδύναμη αριθμητική τους μορφή.

Ανάπτυξη Μοντέλου. Μόλις το σύνολο χαρακτηριστικών είναι έτοιμο, πρέπει να αναπτύξουμε ένα κατάλληλο μοντέλο που να μπορεί να εκπαιδευτεί για να αποκτήσει γνώση από τα δεδομένα. Αυτά τα μοντέλα είναι γενικά στατιστικά, βασισμένα σε μηχανική μάθηση, βασισμένα σε βαθιά μάθηση ή βασισμένα σε ενισχυτική μάθηση. Στην περίπτωσή μας, θα δημιουργήσουμε ένα μοντέλο που θα είναι ικανό να εξάγει συναισθήματα από αριθμητικούς πίνακες.

Εκτίμηση- Αξιολόγηση μοντέλου. Μετά την ανάπτυξη ενός μοντέλου, είναι απαραίτητο να γίνει συγκριτική αξιολόγηση. Αυτή η διαδικασία συγκριτικής αξιολόγησης είναι γνωστή ως αξιολόγηση μοντέλου. Σε αυτό το βήμα, θα αξιολογήσουμε την απόδοση του μοντέλου μας συγκρίνοντάς το με άλλα.

Εφαρμογή μοντέλου. Αυτό είναι το τελικό στάδιο για τα περισσότερα βιομηχανικά έργα NLP. Σε αυτό το στάδιο, τα μοντέλα τίθενται σε παραγωγή. Είτε ενσωματώνονται σε ένα υπάρχον σύστημα είτε δημιουργούνται νέα προϊόντα διατηρώντας αυτό το μοντέλο ως βάση.

3.4.3 Μοντέλο ανάλυσης συναισθήματος (sentiment analysis)

Η πρόβλεψη συναισθήματος στην αγορά προϊόντων και υπηρεσιών, για την αξιολόγηση της ανταγωνιστικής θέσης μίας εταιρείας ή των τάσεων ενός κλάδου, η πρόβλεψη του αντίκτυπου των ειδήσεων που παράγονται και άλλες τέτοιου είδους δραστηριότητες, υποβοηθούνται από την έρευνα και εφαρμογή των μοντέλων που ειδικεύονται στην ανάλυση συναισθήματος. Το κείμενο στο οποίο θα εκπαιδευτεί το μοντέλο, είναι από το αρχείο imdb και περιλαμβάνει κριτικές χρηστών από παρακολούθηση ταινίας. Έχει γίνει η βασική προεργασία σε όλες τις προτάσεις, Οι κριτικές έρχονται χαρακτηρισμένες (labeled) σε θετικές ή αρνητικές. Φορτώνουμε τα δεδομένα και εκπαιδεύουμε τη μηχανή με κάποια υψηλή ακρίβεια, στην αναγνώριση συναισθήματος των χρηστών, Στο τέλος, ελέγχουμε την αποτελεσματικότητα του μοντέλου, σε τυχαίες προτάσεις.

NLP model

```
# Εισάγουμε τις απαραίτητες βιβλιοθήκες

import tensorflow as tf
from tensorflow import keras
```

```
from keras import layers
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Embedding, Conv1D, GlobalMaxPooling1D
from keras.datasets import imdb

# καθορισμός παραμέτρων

max_features = 5000
maxlen = 400
batch_size = 32
embedding_dims = 50
filters = 250
hidden_dims = 250
kernel_size = 3
epochs = 2
import numpy as np
np_load_old = np.load

# τροποποίηση προεπιλεγμένων παραμέτρων

np.load = lambda *a,**k: np_load_old(*a, allow_pickle = True)
print('Loading data...')
# φορτώνουμε τα δεδομένα από το αρχείο imdb, 25000 για εκπαίδευση

# και άλλα τόσα για αξιολόγηση

(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)

print(len(x_train), 'train sequences')
print(len(x_test), 'test sequences')
def get_fixed_word_to_id_dict():
  INDEX_FROM=3

  word_to_id = keras.datasets.imdb.get_word_index()
  word_to_id = {k:(v+INDEX_FROM) for k,v in word_to_id.items()}
  word_to_id[" "] = 0
  word_to_id["<START>"] = 1
  word_to_id["<UNK"] = 2
  return word_to_id

# συνάρτηση που μετατρέπει τις προτάσεις του κειμένου σε αριθμητικούς πίνακες

def decode_to_sentence(data_point):
  NUM_WORDS= 1000
# χρησιμοποιούμε τις 1000 πρώτες λέξεις
```

```
  word_to_id = get_fixed_word_to_id_dict()
  word_to_id = {value:key for key, value in word_to_id.items()}
  return ' '.join(word_to_id[id] for id in data_point)

# παράδειγμα αποκωδικοποίησης  πρότασης

print(decode_to_sentence(x_train[0]))
print(y_train[0])
# εμφανίζει το συναίσθημα # παράδειγμα αποκωδικοποίησης της  πρότασης
# αν είναι 1 είναι θετικό , αν 0 αρνητικό


def encode_sentence(sent):
  # τυπώνει την πρόταση tokenized
  encoded = []
  word_to_id = get_fixed_word_to_id_dict()
  for w in sent.split(' '):
    if w in word_to_id :
      encoded.append(word_to_id[w])
    else:
      encoded.append(2)
  return encoded

print("Pad sequences (samples x time)")
x_train = sequence.pad_sequences(x_train, maxlen=maxlen)
x_test = sequence.pad_sequences(x_test, maxlen=maxlen)

print("Build model...")

model = Sequential()

# ξεκινάμε με ένα αποτελεσματικό επίπεδο ενσωμάτωσης
# που αντιστοιχίζει τους δείκτες λεξιλογίου μας στις διαστάσεις embedding_dims
model.add(Embedding(max_features, embedding_dims, input_length=maxlen))
model.add(Dropout(0.2))
# προσθέτουμε ένα συνελικτικό στρώμα όπου εκπαιδεύσουμε τα φίλτρα
, kernel_size, padding='valid', activation= 'relu', strides=1))
# maxpooling
model.add(GlobalMaxPooling1D())
# προσθέτουμε ένα κρυφό στρώμα
```

```
model.add(Dense(hidden_dims))
model.add(Dropout(0.2))
model.add(Activation('relu'))

# προβάλλουμε ένα ενιαίο στρώμα εξόδου και το ενεργοποιούμε με σιγμοειδή συνάρτηση


model.add(Dense(1))
model.add(Activation('sigmoid'))
model.summary()
model.compile(loss='binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
history = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs)
model.evaluate(x_test, y_test)
predictions = model.predict(x_test)
sentiment= ['Negative' if i<= 0.5   else 'Positive' for i in predictions]
print("Η ακρίβεια που επιτυγχάνει το μοντέλο κατά την εκπαίδευση :")
print(history.history['accuracy'])
```

```
Build model...
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 400, 50)           250000

 dropout (Dropout)           (None, 400, 50)           0

 conv1d (Conv1D)             (None, 398, 250)          37750

 global_max_pooling1d (Globa  (None, 250)              0
 lMaxPooling1D)

 dense (Dense)               (None, 250)               62750

 dropout_1 (Dropout)         (None, 250)               0

 activation (Activation)     (None, 250)               0

 dense_1 (Dense)             (None, 1)                 251

 activation_1 (Activation)   (None, 1)                 0

=================================================================
Total params: 350,751
Trainable params: 350,751
Non-trainable params: 0
_________________________________________________________________
Epoch 1/2
782/782 [==============================] - 79s 99ms/step - loss: 0.4111 - accuracy: 0.7938
Epoch 2/2
782/782 [==============================] - 79s 100ms/step - loss: 0.2320 - accuracy: 0.9059
782/782 [==============================] - 19s 25ms/step - loss: 0.2578 - accuracy: 0.8954
Η ακρίβεια που επιτυγχάνει το μοντέλο κατά την εκπαίδευση :
[0.7938399910926819, 0.9059200286865234]
```

```
# δοκιμή του μοντέλου σε τυχαίες προτάσεις

data_point_to_show = 1
print(decode_to_sentence(x_test[data_point_to_show]), "--", sentiment[data_point_to_show])
test_sentences = []
test_sentence = " I do not like this at all"
test_sentence = encode_sentence(test_sentence)
test_sentences.append(test_sentence)
test_sentence = " i love it"
test_sentence = encode_sentence(test_sentence)
test_sentences.append(test_sentence)
test_sentence = "  i am satisfied about it "
test_sentence = encode_sentence(test_sentence)
test_sentences.append(test_sentence)
test_sentences = sequence.pad_sequences(test_sentences, maxlen=maxlen)
test_sentences.shape
predictions = model.predict(x_test)
sentiment= ['Negative' if i<= 0.5 else 'Positive' for i in predictions]
for i in range(test_sentences.shape[0]):
    print(decode_to_sentence(test_sentences[i]), "--", sentiment[i])

# η έξοδος
<UNK do not like this at all -- Negative
   <UNK i love it -- Positive
<UNK i am satisfied about it <UNK -- Positive
```

Κεφάλαιο 4ο

Πλατφόρμα ανάπτυξης λογισμικού Anaconda

Το Anaconda θεωρείται η πιο ολοκληρωμένη σουίτα για την επιστήμη δεδομένων σε Python/R και μία από τις πιο ολοκληρωμένες για την αποτελεσματική διαχείριση έργων σχετικών με την τεχνητή νοημοσύνη (ML/DL). Παρέχει έναν μεγάλο αριθμό λειτουργιών που μας επιτρέπουν να αναπτύξουμε εφαρμογές με πιο αποτελεσματικό, ταχύτερο και ευκολότερο τρόπο. Η διανομή που μας ενδιαφέρει περισσότερο είναι η Anaconda individual, με το Anaconda Navigator και το Conda, ως βασικά περιβάλλοντα ανάπτυξης των εφαρμογών μας. Στον ιστότοπο https://www.anaconda.com/ η ομάδα του Anaconda, έχει αναπτύξει διάφορες παραλλαγές ως προϊόντα, για δωρεάν ή με κόστος, χρήση της σουίτας.

4.1 Παραλλαγές Anaconda

Anaconda Distribution

Το Anaconda Distribution είναι η βασική διανομή του Anaconda σε Python/R, η οποία ενσωματώνει μία συλλογή με περισσότερα από 7.500+ πακέτα ανοιχτού κώδικα, έναν διαχειριστή πακέτων και έναν περιβάλλοντος. Το Anaconda Distribution είναι αγνωστικό ως προς την πλατφόρμα (platform-agnostic), που σημαίνει ότι μπορούμε να κάνουμε χρήση σε οποιοδήποτε περιβάλλον Windows, macOS ή Linux. Είναι επίσης δωρεάν (Anaconda individual) για εγκατάσταση και προσφέρει δωρεάν υποστήριξη.

Αφού εγκαταστήσουμε το Anaconda ή το Miniconda, εάν προτιμήσουμε μία επιφάνεια εργασίας γραφικών χρήστη (GUI) τότε συνίσταται να χρησιμοποιήσουμε το Navigator . Εάν θέλουμε άμεση απόκριση του Anaconda (όπως τερματικό σε Linux ή macOS), χρησιμοποιούμε το conda. Μπορούμε επίσης να κάνουμε εναλλαγή μεταξύ τους.

Έχουμε τη δυνατότητα να εγκαταστήσουμε, να αφαιρέσουμε ή να ενημερώσουμε οποιοδήποτε πακέτο Anaconda με μερικά κλικ στο Navigator ή με μία μόνο εντολή conda στο Anaconda Prompt (τερματικό σε Linux ή macOS).

Προηγούμενες εκδόσεις του Anaconda είναι διαθέσιμες στο αρχείο. Για μία λίστα με τα πακέτα που περιλαμβάνονται σε κάθε προηγούμενη έκδοση, ανατρέχουμε στην ενότητα, "παλιές

λίστες πακέτων". Το Anaconda2 περιλαμβάνει Python 2.7 και το Anaconda3 Python 3.7. Ωστόσο, δεν έχει σημασία ποιο θα κατεβάσουμε, γιατί μπορούμε να δημιουργήσουμε νέα περιβάλλοντα που περιλαμβάνουν οποιαδήποτε έκδοση της Python μαζί με το conda.

Anaconda Professional

Παλαιότερα γνωστή ως Commercial Edition

Το Anaconda Professional είναι μία από τις πιο δημοφιλής εκδόσεις διανομής και διαχείρισης πακέτων ανοιχτού κώδικα στον κόσμο, βελτιστοποιημένη για εμπορική χρήση.

Βασικά χαρακτηριστικά:

- Περισσότερα από 7.500 πακέτα επιστήμης δεδομένων / μηχανικής μάθησης που έχουν δημιουργηθεί από την Anaconda
- Ασφαλής πρόσβαση στο χώρο αποθήκευσης εμπορικών πακέτων μας
- Δυνατότητα αξιοποίησης λογισμικού mirroring για τη δημιουργία αντιγράφων του αποθετηρίου εμπορικών πακέτων (μόνο άδεια τοποθεσίας)
- Εταιρική κλίμακα και διαθεσιμότητα
- Συμβατό για εμπορική χρήση σύμφωνα με τους Όρους Παροχής Υπηρεσιών της Anaconda

Anaconda Business

Το Anaconda Business εξουσιοδοτεί τους οργανισμούς να επιβάλλουν πολιτικές ασφάλειας για λογισμικό ανοιχτού κώδικα στο περιβάλλον Anaconda cloud, χωρίς να θυσιάζεται η ταχύτητα.

Anaconda Server

Παλαιότερα γνωστό ως Team Edition

Ο διακομιστής Anaconda αποτελεί ένα αποθετήριο τελευταίας γενιάς για το περιεχόμενο του Anaconda. Με υποστήριξη για όλα τα μεγάλα λειτουργικά συστήματα, το αποθετήριο χρησιμεύει ως ο κεντρικός πόρος Conda, PyPI και CRAN.

Anaconda Enterprise Edition

Το Anaconda Enterprise είναι μία έτοιμη για επιχειρήσεις, ασφαλής και επεκτάσιμη πλατφόρμα επιστήμης δεδομένων που εξουσιοδοτεί τις ομάδες να χειρίζονται και να συνεργάζονται για την ανάπτυξη έργων.

Το Anaconda Enterprise, παρέχει τα εξής:

- Ανάπτυξη : δυνατότητα ML/AI σε ένα κεντρικό περιβάλλον ανάπτυξης που κλιμακώνεται από φορητούς υπολογιστές σε χιλιάδες κόμβους
- Διαχείριση : Πλήρης αναπαραγωγιμότητα από φορητό υπολογιστή, με δυνατότητα ρύθμισης παραμέτρων ελέγχου πρόσβασης
- Αυτοματοποίηση : Μοντέλο εκπαίδευσης και ανάπτυξης σε επεκτάσιμη υποδομή που βασίζεται σε conteiners

Anaconda.org

Το Anaconda.org είναι μία υπηρεσία διαχείρισης πακέτων που διευκολύνει την εύρεση, πρόσβαση, αποθήκευση και κοινή χρήση δημόσιων σημειωματάριων και περιβαλλόντων, καθώς και πακέτων conda και PyPI. Το Anaconda.org φιλοξενεί εκατοντάδες χρήσιμα πακέτα Python, σημειωματάρια, έργα και περιβάλλοντα για μία μεγάλη ποικιλία εφαρμογών. Δεν χρειάζεται να συνδεθούμε, ούτε καν να έχουμε λογαριασμό στο Anaconda.org, για να αναζητήσουμε δημόσια πακέτα, να τα κατεβάσουμε και να τα εγκαταστήσουμε. Μπορούμε να δημιουργήσουμε νέα πακέτα conda χρησιμοποιώντας το conda-build και στη συνέχεια, να ανεβάσουμε τα πακέτα στο Anaconda.org ώστε να τα μοιραστούμε με άλλους ή να αποκτήσουμε πρόσβαση από οπουδήποτε. Για τους προγραμματιστές, το Anaconda.org έχει σχεδιαστεί για να διευκολύνει την ανάπτυξη, τη κυκλοφορία και τη συντήρηση λογισμικού παρέχοντας ευρεία υποστήριξη διαχείρισης πακέτων. Το Anaconda.org επιτρέπει δωρεάν δημόσια φιλοξενία πακέτων, καθώς και κανάλια πακέτων, παρέχοντας μία ευέλικτη και επεκτάσιμη υπηρεσία για ομάδες και οργανισμούς όλων των μεγεθών.

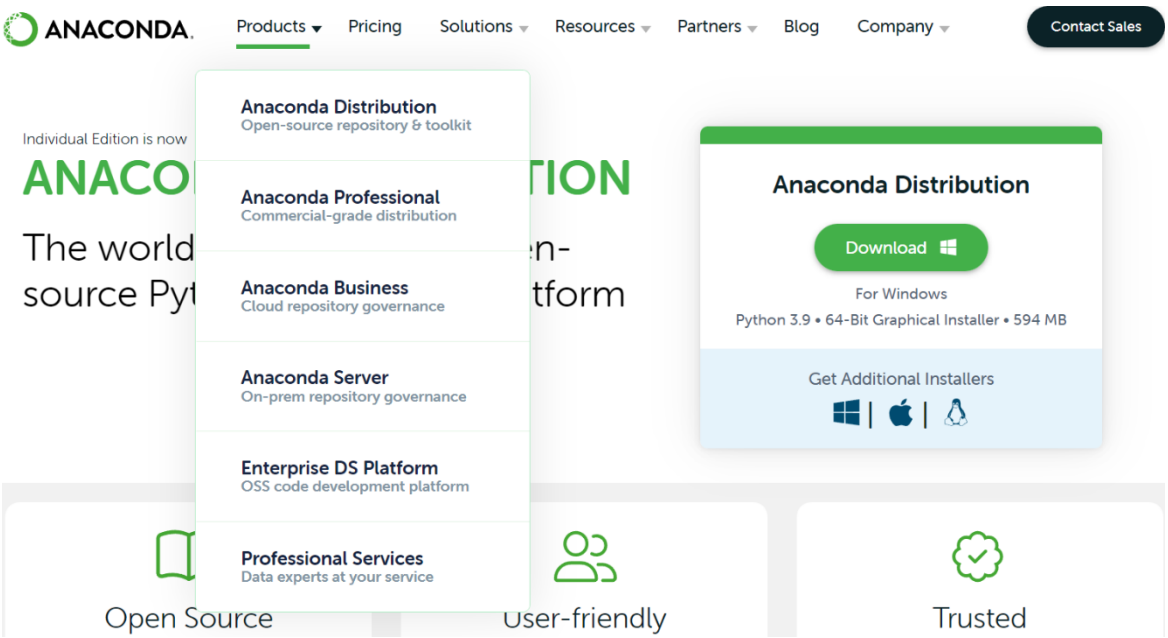
Anaconda Embedded

Το Anaconda Embedded, μας επιτρέπει να τροφοδοτούμε προϊόντα και υπηρεσίες, με μία από τις δημοφιλέστερες διανομές πακέτων ανοιχτού κώδικα στον κόσμο. Το Anaconda Embedded δίνει τη δυνατότητα, ως πάροχος προϊόντων ή υπηρεσιών, να προσφέρουμε μία απρόσκοπτη διεπαφή Python για τους πελάτες. Όλοι οι ενσωματωμένοι συνεργάτες λαμβάνουν πρόσβαση στο premium αποθετήριο, τους ειδικούς και τους προγραμματιστές της Anaconda, καθώς και πρόσθετα οφέλη όπως SLA, co-marketing και προσαρμοσμένες ευκαιρίες ανάπτυξης.

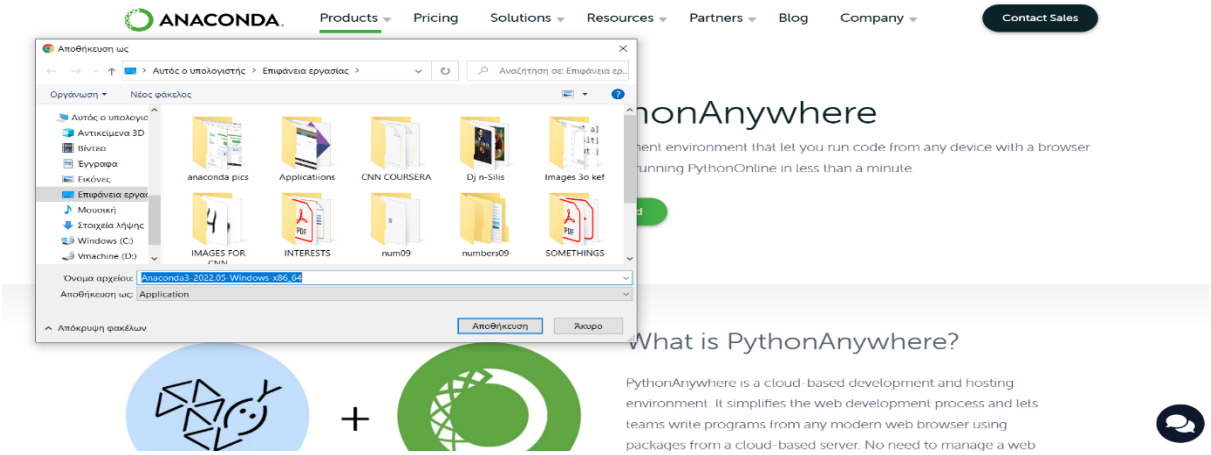
4.2 Εγκατάσταση Anaconda individual distribution

Για την εγκατάσταση, οδηγούμαστε στον ιστότοπο του Anaconda, μέσω της διεύθυνσης https://www.anaconda.com/ και επιλέγουμε από τα products το Anaconda Distribution, εμφανίζεται το παράθυρο της individual edition και πατάμε Download για Windows. Για

MacOS & Linux, επιλέγουμε τα αντίστοιχα εικονίδια κάτω από “Get Additional Installers” και μας μεταφέρει στο τέλος της σελίδας, όπου βρίσκονται οι οδηγοί για αυτά τα λειτουργικά συστήματα.

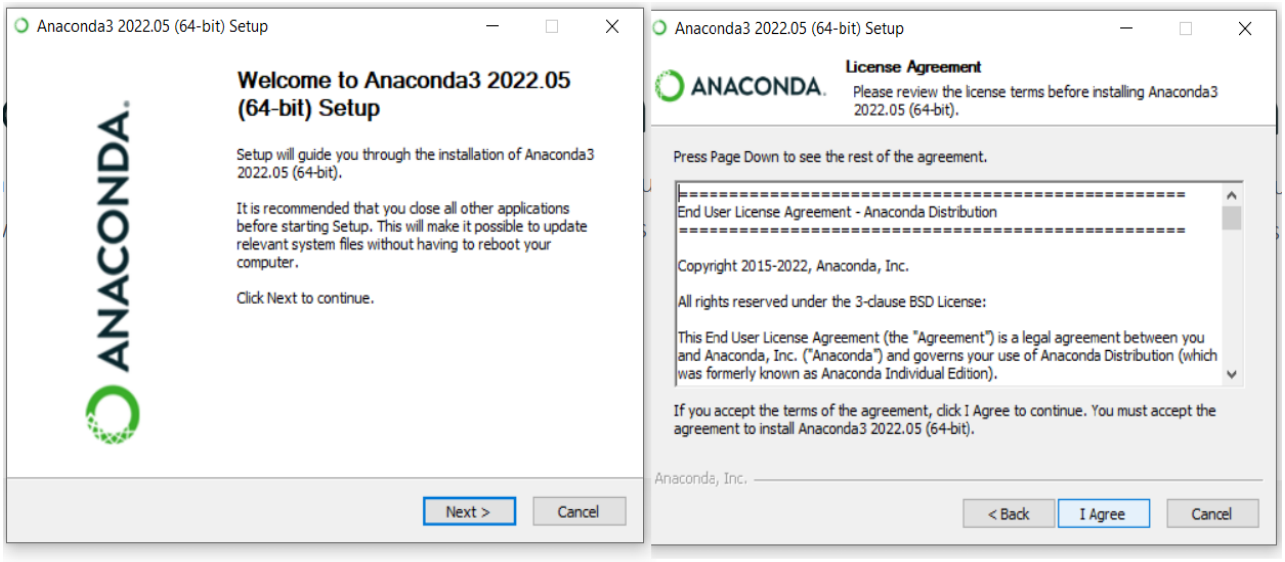


Αποθηκεύουμε ως application

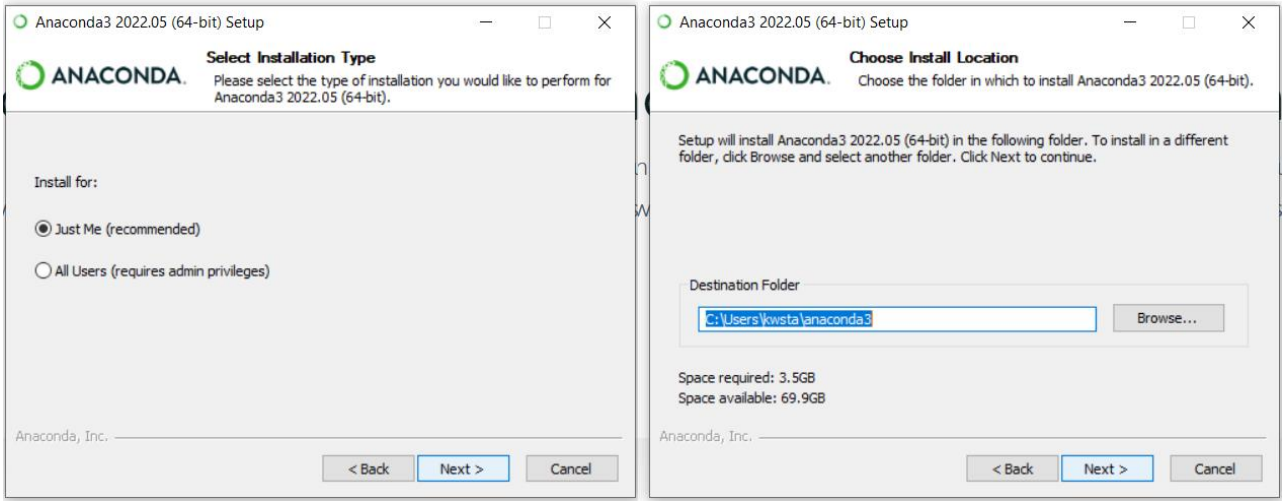


Όταν ολοκληρωθεί η διαδικασία, τρέχουμε το executable αρχείο που περιέχει τις οδηγίες εγκατάστασης του Anaconda, οι οποίες περιγράφονται στις διαδοχικές εικόνες που ακολουθούν.

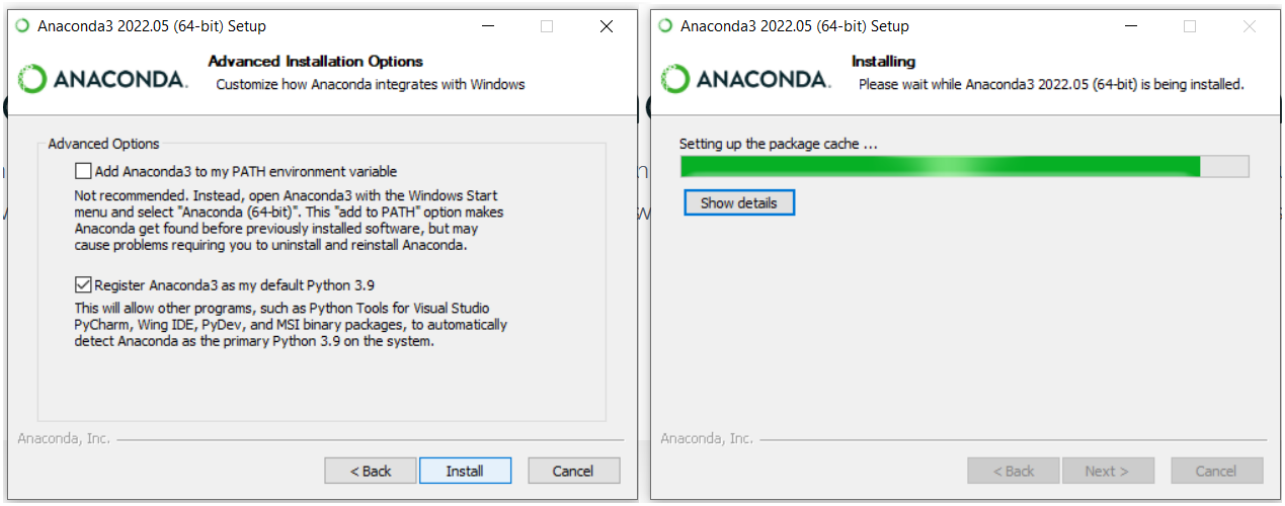
Επιλέγουμε Next και αφού συμφωνούμε με τους όρους, επιλέγουμε I Agree



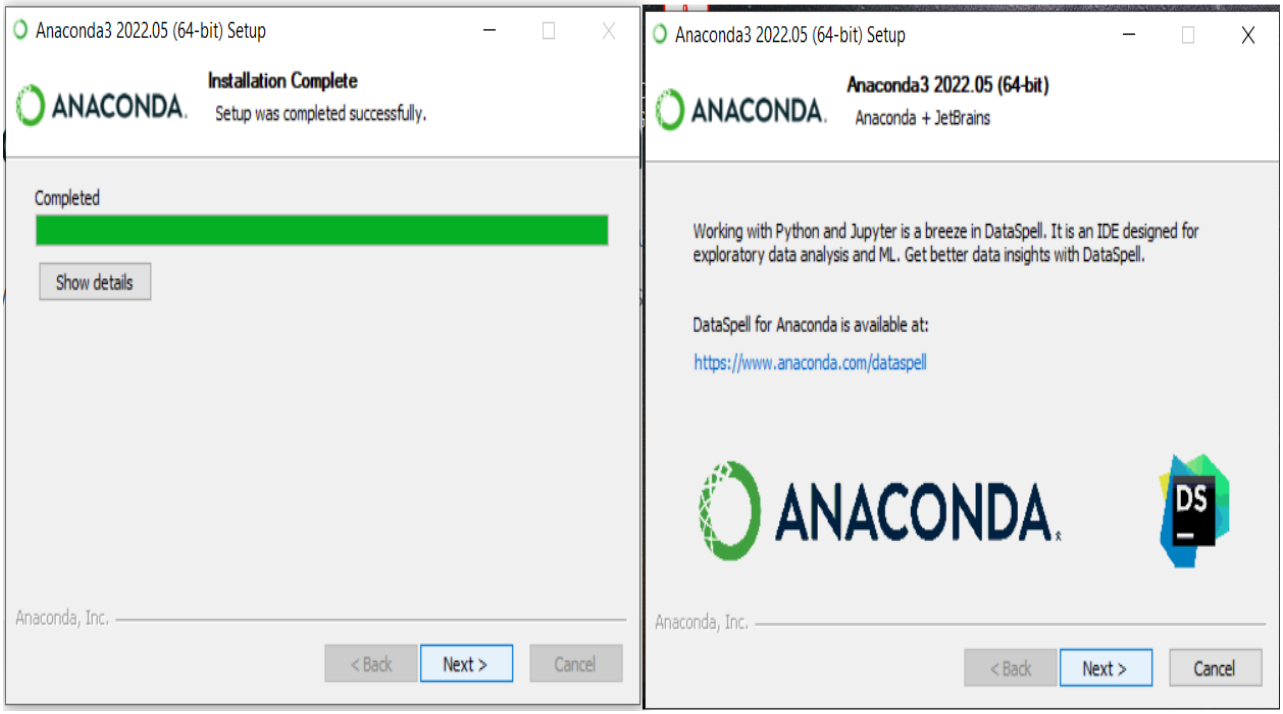
Στην 1η καρτέλα τικάρουμε Just Me & Next, στην 2η επιλέγουμε που θα αποθηκεύσουμε το αρχείο Anaconda3, (ο απαιτούμενος αποθηκευτικός χώρος είναι 3.5GB) και πατάμε Next



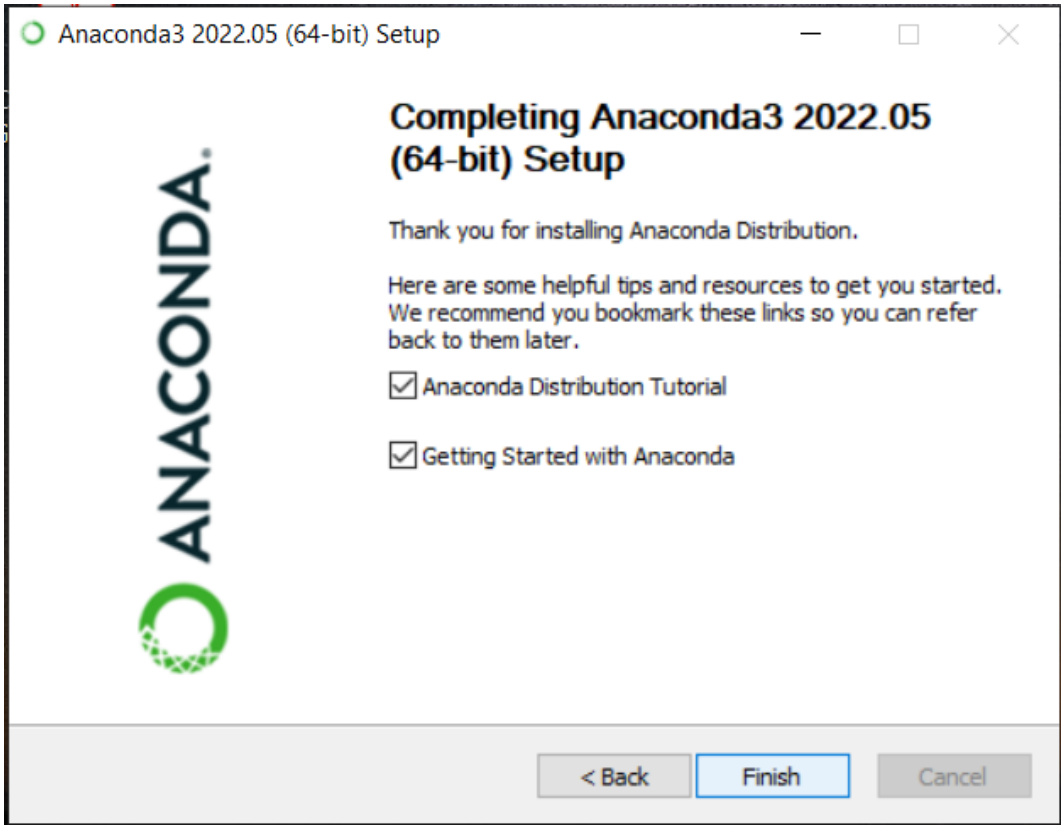
Ξεκινά η εγκατάσταση του Anaconda. Στην 1η καρτέλα τικάρουμε τη δεύτερη επιλογή και Install, ενώ στην 2η βλέπουμε πως προχωράει η εγκατάσταση.



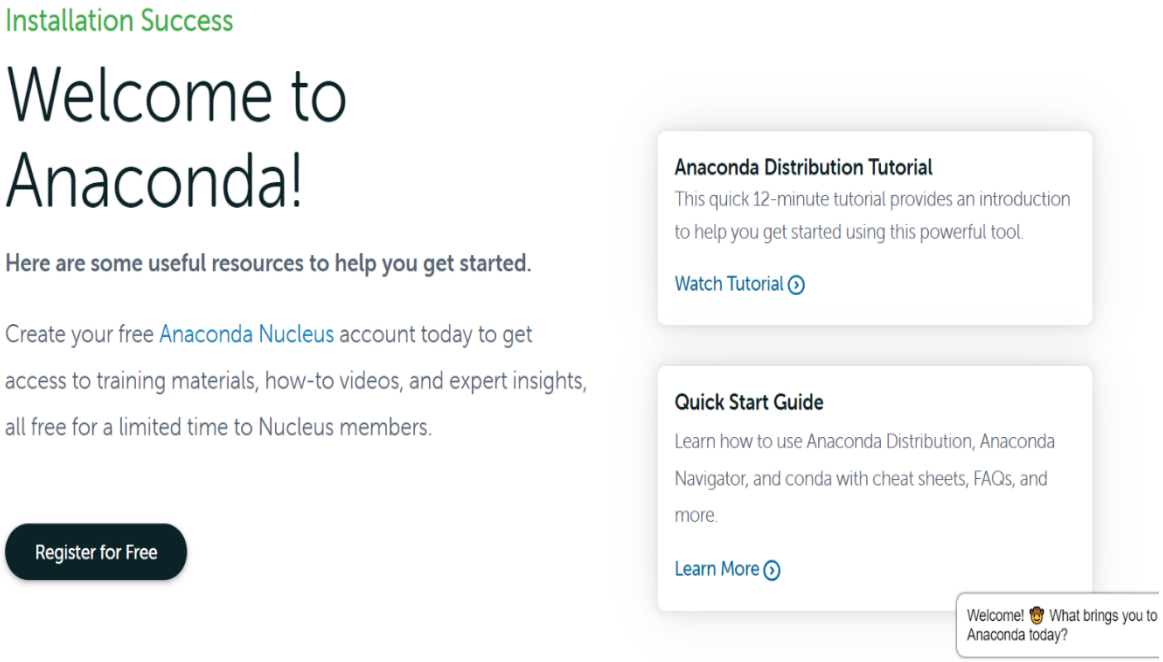
Αφού ολοκληρωθεί η εγκατάσταση επιλέγουμε Next και στις δύο επόμενες καρτέλες



Μετά την ολοκλήρωση τικάρουμε τις επιλογές και επιλέγουμε Finish



Τέλος, αν όλα πήγαν καλά, μας βγάζει στη σελίδα του Anaconda. Μπορούμε αν θέλουμε, να πάρουμε κάποια μαθήματα για το περιεχόμενο της διανομής που εγκαταστήσαμε.



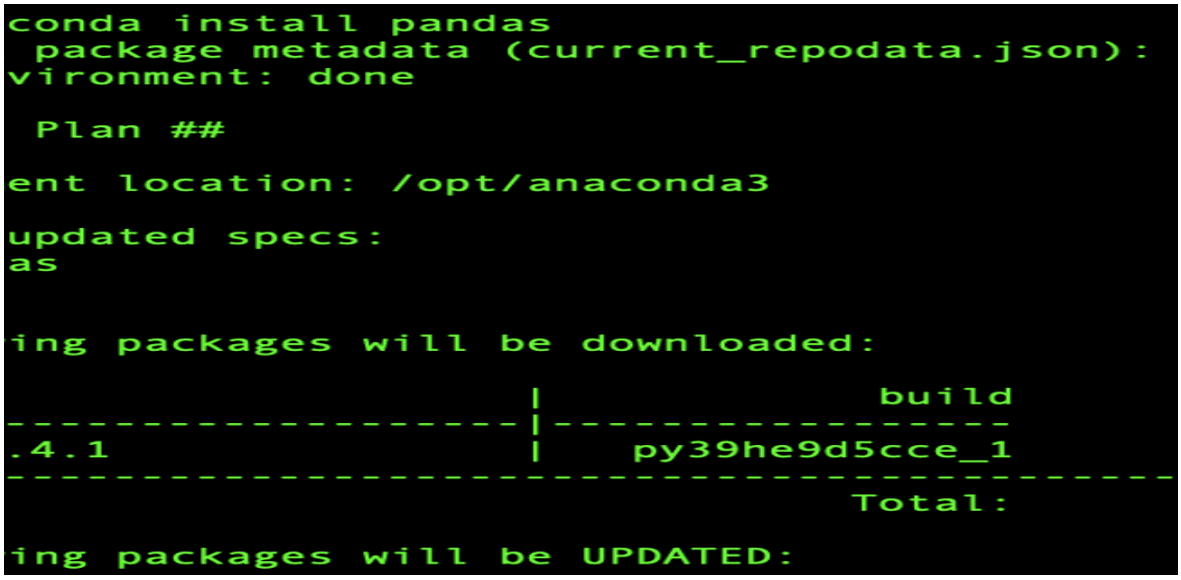
4.3 Conda – Anaconda Navigator

Μία από τις προκλήσεις που εμφανίζονται κατά την εκτέλεση εργασιών πάνω σε θέματα που αφορούν την επιστήμη δεδομένων και τη μηχανική μάθηση, είναι η ύπαρξη πολλών πακέτων λογισμικού που πρέπει να εγκατασταθούν. Τα πακέτα, ενσωματώνουν στοιχεία τα οποία έχουν γραφεί σε διαφορετική γλώσσα από αυτές που χρησιμοποιούνται στην πλατφόρμα δηλαδή τις Python & R. Η εγκατάσταση και η εξασφάλιση της συμβατότητας μεταξύ των πακέτων λογισμικού, αποτελεί μία σταθερά, για την ομαλή ροή των εργασιών που έχουμε αναλάβει. Για να εκτελεστούν, πολλά επιστημονικά πακέτα εξαρτώνται από συγκεκριμένες εκδόσεις άλλων πακέτων. Οι επιστήμονες δεδομένων χρησιμοποιούν συχνά πολλαπλές εκδόσεις πολλών πακέτων και χρησιμοποιούν πολλαπλά περιβάλλοντα για να διαχωρίσουν αυτές τις διαφορετικές εκδόσεις. Το Anaconda, βοηθά τους επιστήμονες δεδομένων να διασφαλίσουν ότι η έκδοση κάθε πακέτου έχει όλες τις εξαρτήσεις που απαιτεί και λειτουργεί σωστά. Η ομάδα του Anaconda, έχει ενσωματώσει χιλιάδες βιβλιοθήκες με ευρεία χρήση, σε Python/R, αλλά και σε άλλες γλώσσες προγραμματισμού. Η εγκατάσταση της δωρεάν έκδοσης του Anaconda (individual edition), φέρει περισσότερα από 200 προεγκατεστημένα πακέτα λογισμικού, έτοιμα προς χρήση, εύκολα στην αναβάθμιση, καθώς και τη δυνατότητα εγκατάστασης νέων πακέτων. Επιπλέον είναι ενσωματωμένα δύο συστήματα διαχείρισης πακέτων και περιβάλλοντος, το Conda και το Anaconda Navigator.

4.3.1 Conda

Το Conda είναι ένα πακέτο ανοιχτού κώδικα και σύστημα διαχείρισης περιβάλλοντος που τρέχει σε Windows, macOS και Linux. Το Conda εγκαθιστά, εκτελεί και ενημερώνει γρήγορα τα πακέτα και τις εξαρτήσεις τους. Επίσης, δημιουργεί, αποθηκεύει, φορτώνει και αλλάζει εύκολα περιβάλλοντα στον τοπικό υπολογιστή. Δημιουργήθηκε για προγράμματα Python, αλλά μπορεί να διανείμει λογισμικό για οποιαδήποτε γλώσσα. Κάτι παρόμοιο θα λέγαμε ότι κάνει το Pip το οποίο είναι ένας διαχειριστής πακέτων που διευκολύνει την εγκατάσταση, την αναβάθμιση και την απεγκατάσταση πακέτων python . Λειτουργεί με εικονικά περιβάλλοντα python και επικεντρώνεται γύρω από την Python, παραβλέποντας εξαρτήσεις βιβλιοθήκης εκτός Python, όπως HDF5, MKL, LLVM που δεν έχουν αρχείο εγκατάστασης στον πηγαίο κώδικα τους. Το Conda στοχεύει κυρίως να κάνει περισσότερα από ότι η pip: δηλαδή να χειρίζεται εξαρτήσεις βιβλιοθήκης *εκτός* των πακέτων Python καθώς και των ίδιων των πακέτων Python. Το Conda δημιουργεί επίσης ένα εικονικό περιβάλλον και είναι γραμμένο εξ

ολοκλήρου σε Python, γεγονός που καθιστά ευκολότερη τη χρήση σε εικονικά περιβάλλοντα Python. Επιπλέον, μπορούμε να χρησιμοποιήσουμε το Conda για βιβλιοθήκες C, πακέτα R, πακέτα Java και ούτω καθεξής. Επίσης το εργαλείο conda build δημιουργεί πακέτα από τη πηγή και το conda install εγκαθιστά βιβλιοθήκες από ενσωματωμένα πακέτα conda, όπως φαίνεται στην εικόνα.



4.3.2 Anaconda Navigator

Το Anaconda Navigator, είναι ένα γραφικό περιβάλλον εργασίας χρήστη (GUI), που περιλαμβάνεται στη διανομή Anaconda, το οποίο μας επιτρέπει τη χρήση εφαρμογών, να διαχειριζόμαστε εύκολα πακέτα, περιβάλλοντα και κανάλια conda χωρίς να χρησιμοποιούμε τη γραμμή εντολών. Το Navigator μπορεί να αναζητήσει πακέτα στο Anaconda.org ή σε ένα τοπικό αποθετήριο Anaconda. Είναι διαθέσιμο για Windows, macOS, Linux και εγκαθίσταται μαζί με την individual edition του Anaconda. Για το linux τρέχει από το terminal, με την εντολή anaconda-navigator, για το mac os βρίσκουμε το εικονίδιο του AN, στο application directory ή στο launchpad, για τα windows, βρίσκουμε στο menu εκκίνησης το anaconda3.

Το Navigator είναι ένας εύκολος τρόπος να εργαζόμαστε με πακέτα και περιβάλλοντα, χωρίς να χρειάζεται να πληκτρολογήσουμε εντολές conda, σε ένα παράθυρο τερματικού. Μπορούμε να το χρησιμοποιήσουμε για να βρούμε τα πακέτα που θέλουμε, να τα εγκαταστήσουμε σε ένα περιβάλλον, να εκτελέσουμε τα πακέτα και να τα ενημερώσουμε, όλα μέσα από το Navigator.

ΣΦΑΙΡΙΚΗ ΕΙΚΟΝΑ

Αυτή η σελίδα περιγράφει τις καρτέλες, τα μενού και τα κουμπιά στο παράθυρο του Anaconda Navigator.

- Online and offline modes
- Home tab
- Environments tab
- Learning tab
- Community tab
- File menu
- Help menu
- Navigator window buttons

Online και offline λειτουργίες

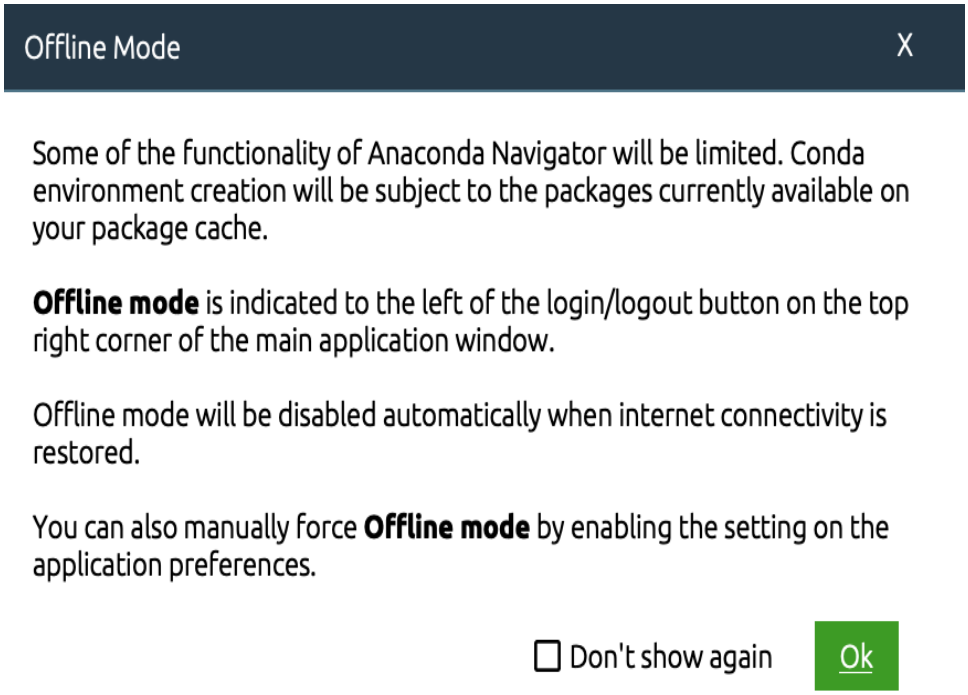
Κανονικά το Navigator χρησιμοποιείται διαδικτυακά για να μπορεί να κατεβάσει και να εγκαταστήσει πακέτα.

Σε λειτουργία online, το Navigator πρέπει να μπορεί να προσεγγίσει αυτούς τους ιστότοπους, επομένως μπορεί να χρειαστεί να συμπεριληφθούν στη λίστα επιτρεπόμενων στις ρυθμίσεις του τείχους προστασίας του δικτύου σας.

- https://repo.anaconda.com
- https://conda.anaconda.org για το conda-forge και άλλα κανάλια στο anaconda.org
- google-public-dns-a.google.com (8.8.8.8:53) για να ελέγξετε τη σύνδεση στο διαδίκτυο με το Google Public DNS

Λειτουργία εκτός σύνδεσης

Εάν το Navigator εντοπίσει ότι η πρόσβαση στο Διαδίκτυο δεν είναι διαθέσιμη, ενεργοποιεί αυτόματα τη λειτουργία εκτός σύνδεσης και εμφανίζει αυτό το μήνυμα:

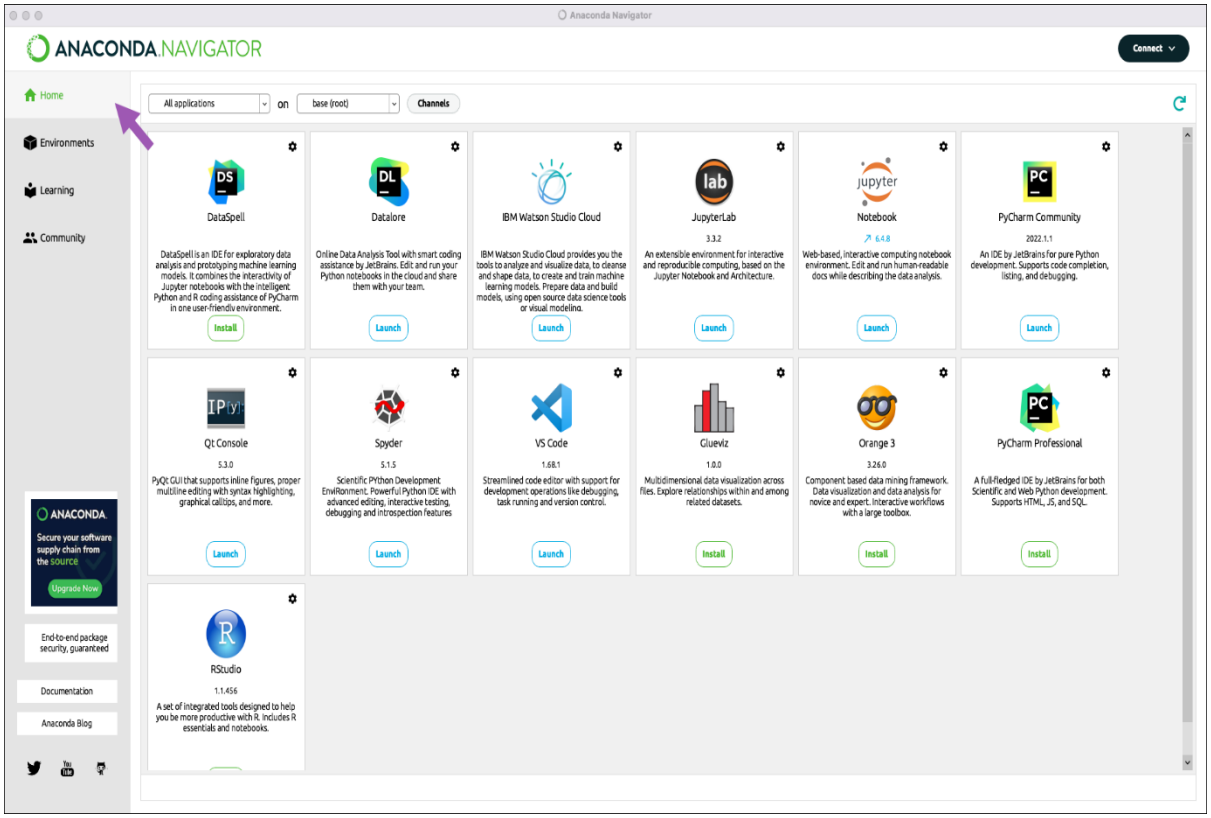


Στο παράθυρο διαλόγου "Προτιμήσεις", επιλέξτε "Ενεργοποίηση λειτουργίας εκτός σύνδεσης" για να εισέλθετε σε λειτουργία εκτός σύνδεσης, ακόμη και αν είναι διαθέσιμη η πρόσβαση στο Διαδίκτυο.

Η χρήση του Navigator σε λειτουργία εκτός σύνδεσης ισοδυναμεί με τη χρήση των εντολών conda της γραμμής εντολών create, install, remove, και update με τη ένδειξη --offline, έτσι ώστε η conda να μην συνδέεται στο διαδίκτυο.

Καρτέλα Home

Η αρχική καρτέλα, που φαίνεται στην παρακάτω εικόνα, εμφανίζει όλες τις διαθέσιμες εφαρμογές που μπορούμε να διαχειριστούμε με το Navigator.



Την πρώτη φορά που ανοίγουμε το Navigator, οι ακόλουθες δημοφιλείς γραφικές εφαρμογές Python έχουν ήδη εγκατασταθεί ή είναι διαθέσιμες για εγκατάσταση:

- JupyterLab
- Jupyter Notebook
- Spyder
- PyCharm
- VSCode
- Glueviz
- Orange 3 App
- RStudio
- Anaconda Prompt (Windows only)
- Anaconda PowerShell (Windows only)

Μπορούμε επίσης να δημιουργήσουμε τις δικές μας εφαρμογές Navigator

Σε κάθε πλαίσιο εφαρμογής, μπορούμε να :

- Εκκινήσουμε την εφαρμογή – Κάνουμε κλικ στο κουμπί Εκκίνηση.
- Εγκατάσταση μίας εφαρμογής – Κάνουμε κλικ στο κουμπί Εγκατάστασή της.
- Ενημερώνουμε, αφαιρούμε ή εγκαθιστούμε συγκεκριμένη έκδοση μίας εφαρμογής – Κάνουμε κλικ στο εικονίδιο με το γρανάζι στην επάνω δεξιά γωνία του πλαισίου της εφαρμογής.

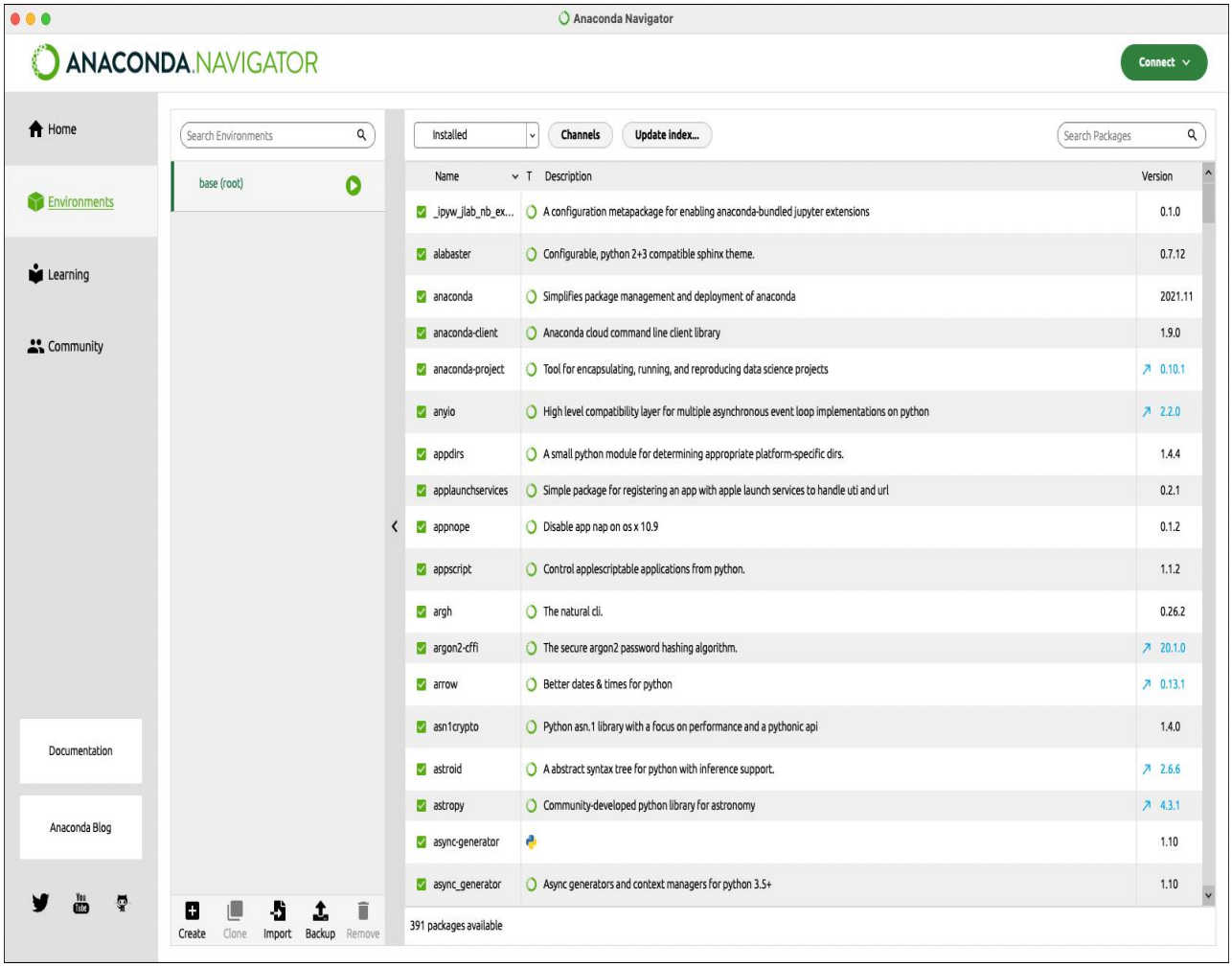
Οι εφαρμογές εγκαθίστανται στο ενεργό περιβάλλον, το οποίο εμφανίζεται στη λίστα "All applications on base (root)". Για να εγκαταστήσουμε μία εφαρμογή σε ένα συγκεκριμένο περιβάλλον, επιλέγουμε πρώτα το περιβάλλον στη λίστα και μετά κάνουμε κλικ στο κουμπί **Install** (Εγκατάσταση της εφαρμογής). Μπορούμε επίσης να δημιουργήσουμε ένα νέο περιβάλλον στην καρτέλα **Environments** και στη συνέχεια, να επιστρέψουμε στην καρτέλα **Home** για να εγκαταστήσουμε πακέτα στο νέο περιβάλλον.

Πώς μπορώ να εκτελέσω κώδικα με το Navigator;

Ο πιο απλός τρόπος είναι με το Spyder. Από την καρτέλα Navigator Home, κάνουμε κλικ στο Spyder και γράφουμε και εκτελούμε τον κώδικά μας. Μπορούμε επίσης να χρησιμοποιήσουμε το Jupyter notebook με τον ίδιο τρόπο. Τα σημειωματάρια Jupyter είναι ένα όλο και πιο δημοφιλές σύστημα που συνδυάζει τον κώδικα, το περιγραφικό κείμενο, την έξοδο, τις εικόνες και τις διαδραστικές διεπαφές, σε ένα ενιαίο αρχείο σημειωματάριου που επεξεργάζεται, προβάλλεται και χρησιμοποιείται σε ένα πρόγραμμα περιήγησης ιστού. Δεν υποστηρίζεται macOS<10.12 για Anaconda Navigator.

Καρτέλα Environments

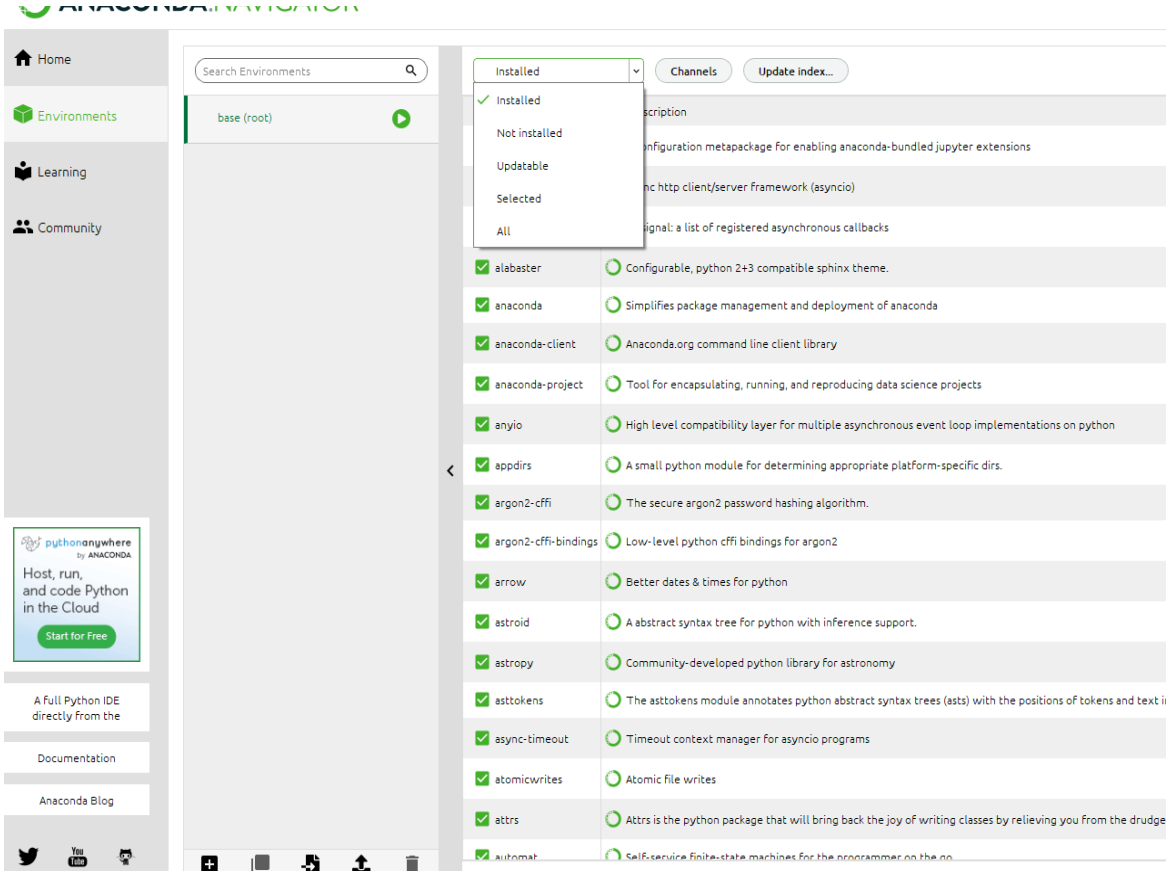
Η καρτέλα **Περιβάλλοντα** μας επιτρέπει να διαχειριζόμαστε εγκατεστημένα περιβάλλοντα, πακέτα και κανάλια.



Η αριστερή στήλη παραθέτει τα περιβάλλοντά μας (base-root). Κάντε κλικ σε ένα περιβάλλον για να το ενεργοποιήσετε.

Με το Navigator όπως και με το conda, μπορούμε να δημιουργήσουμε, να εξάγουμε, να καταγράψουμε, να αφαιρέσουμε και να ενημερώσουμε περιβάλλοντα που έχουν εγκατεστημένες διαφορετικές εκδόσεις Python ή/και πακέτα. Η εναλλαγή ή η μετακίνηση μεταξύ περιβαλλόντων ονομάζεται ενεργοποίηση του περιβάλλοντος. Μόνο ένα περιβάλλον είναι ενεργό ανά πάσα στιγμή. Για περισσότερες πληροφορίες, ανατρέξτε στο θέμα Διαχείριση περιβαλλόντων

Η δεξιά στήλη παραθέτει πακέτα στο τρέχον περιβάλλον. Η προεπιλεγμένη προβολή είναι **Installed** (Εγκατεστημένα πακέτα). Για να αλλάξετε τα πακέτα που εμφανίζονται, κάντε κλικ στο βέλος δίπλα στη λίστα και, στη συνέχεια, επιλέξτε Μη εγκατεστημένο, Αναβαθμίσιμο ή Όλα τα πακέτα.

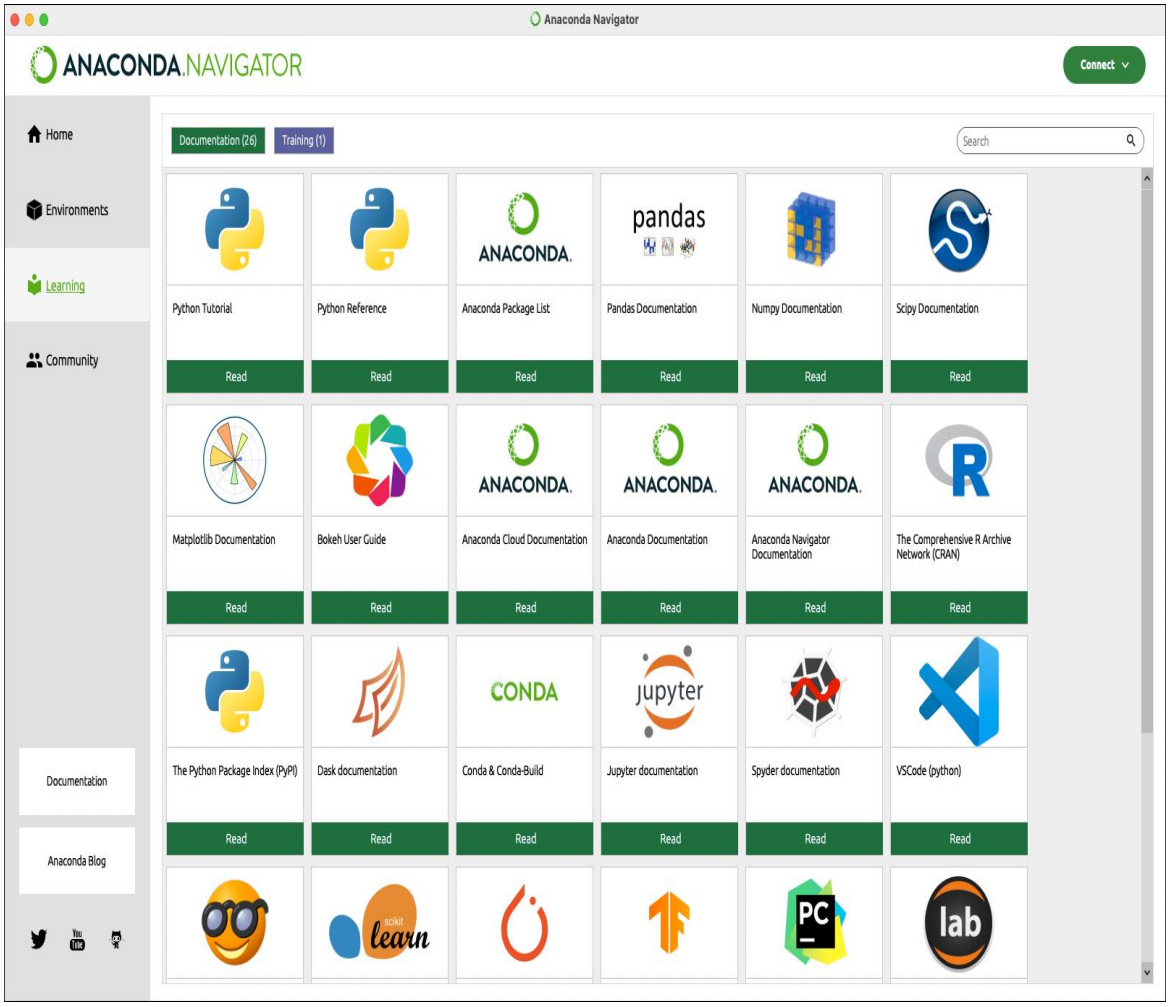


Για περισσότερες πληροφορίες, ανατρέξτε στην ενότητα Διαχείριση πακέτων

Τα κανάλια είναι τοποθεσίες όπου το Navigator ή το conda αναζητά πακέτα. Κάνουμε κλικ στο κουμπί Κανάλια για να ανοίξουμε τη Διαχείριση καναλιών. Για περισσότερες πληροφορίες, ανατρέχουμε στην ενότητα Διαχείριση καναλιών

Καρτέλα Learning

Στην καρτέλα **Εκμάθησης** μπορούμε να μάθουμε περισσότερα για το Navigator, τη πλατφόρμα Anaconda και την επιστήμη ανοιχτών δεδομένων. Κάνουμε κλικ στα κουμπιά Documentation, Training, Webinars ή Video και στη συνέχεια, κάνουμε κλικ σε οποιοδήποτε στοιχείο για να το ανοίξουμε σε ένα παράθυρο του προγράμματος περιήγησης.

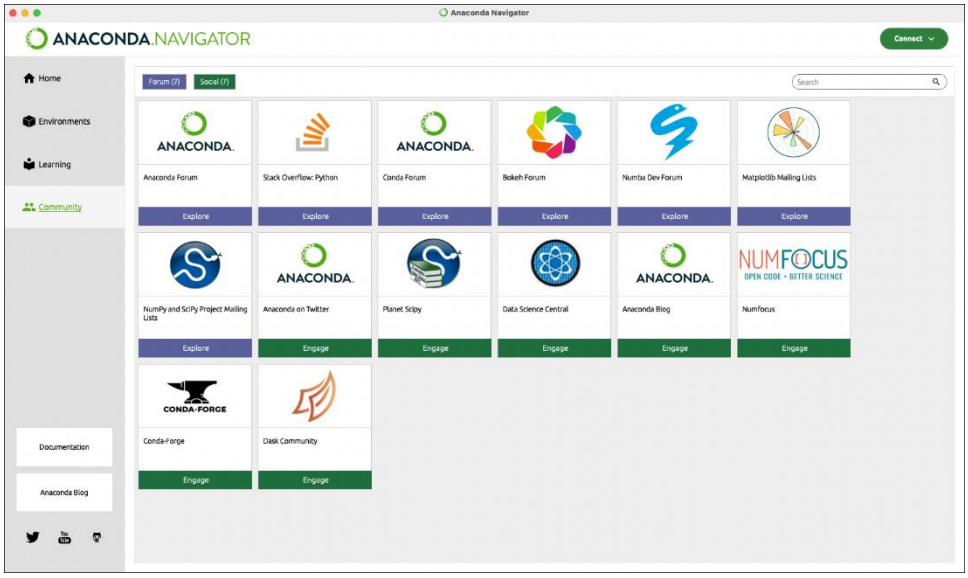


Καρτέλα Community

Στην καρτέλα **Κοινότητα** μπορούμε να μάθουμε περισσότερα για εκδηλώσεις, δωρεάν φόρουμ υποστήριξης και κοινωνική δικτύωση που σχετίζονται με το Navigator. Κάνουμε κλικ στα κουμπιά Εκδηλώσεις, Φόρουμ ή Κοινωνικά και στη συνέχεια, κάνουμε κλικ σε οποιοδήποτε στοιχείο για να το ανοίξουμε σε ένα παράθυρο του προγράμματος περιήγησης.

Υπόδειξη

Για να λάβετε βοήθεια με το Anaconda και το Navigator από την κοινότητα, εγγραφείτε στην κοινότητα Nucleus .



Μενού file

Το μενού **file** του Anaconda Navigator (το όνομα μπορεί να διαφέρει ανάλογα με το λειτουργικό σύστημα) και η μέθοδος εκκίνηση, περιέχει τις ακόλουθες επιλογές:

Πληροφορίες : εμφανίζει πληροφορίες σχετικά με το Navigator, συμπεριλαμβανομένου ενός συνδέσμου για αναφορές σφαλμάτων και αιτήματα για λειτουργίες. Στο Linux αυτό βρίσκεται στο μενού Βοήθεια.

Προτιμήσεις : μας επιτρέπει να ορίσουμε τις προτιμήσεις μας στο Navigator.

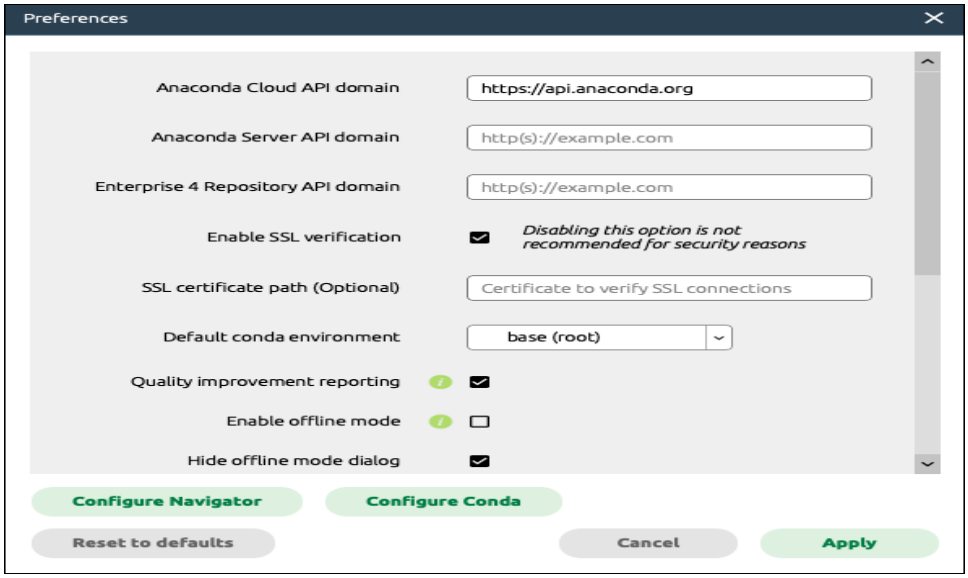
Στο παράθυρο Προτιμήσεις μπορούμε να :

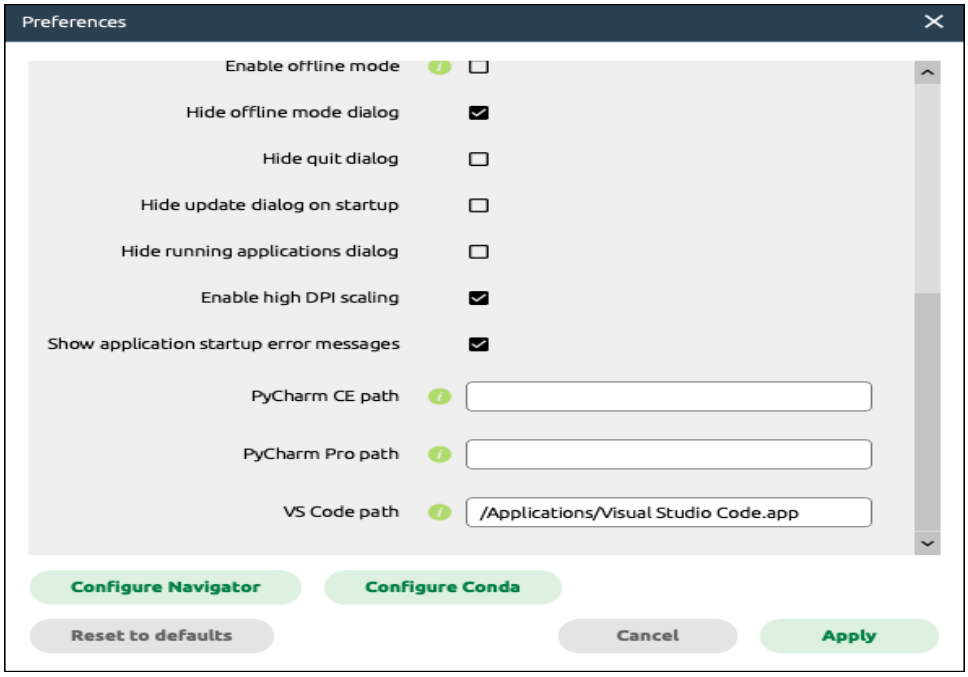
- Αποκρύψουμε το παράθυρο διαλόγου "Έξοδος κατά την έξοδο από το πρόγραμμα".
- Αποκρύψουμε το πλαισίου διαλόγου "Ενημέρωση κατά την εκκίνηση του προγράμματος".
- Αποκρύψουμε τον διαλόγο "Κλείσιμο εφαρμογών που εκτελούνται", το οποίο συνήθως εμφανίζεται κατά την έξοδο από το πρόγραμμα εάν εξακολουθούν να υπάρχουν.
- Ορίσουμε τον τομέα API Anaconda.org εάν πρόκειται να χρησιμοποιήσουμε κανάλια και πακέτα από το anaconda.org.
- Ορίσουμε τον τομέα Anaconda Server API εάν πρόκειται να χρησιμοποιήσουμε κανάλια και πακέτα από το Anaconda Server.

- Ορίσουμε τον τομέα Enterprise 4 Repository API εάν πρόκειται να χρησιμοποιήσουμε κανάλια και πακέτα από τον διακομιστή σας Repo 4.
- Ενεργοποιήσουμε ή απενεργοποιήσουμε την επαλήθευση SSL.
- Προαιρετικά ορίζουμε ένα πιστοποιητικό για την επαλήθευση των συνδέσεων SSL.
- Ενεργοποιούμε την επιλογή παροχής προσωπικών μη αναγνωρίσιμων πληροφοριών για τη βελτίωση του προϊόντος.
- Ενεργοποίηση ή απενεργοποίηση της λειτουργίας εκτός σύνδεσης.
- Αποκρύψουμε το πλαίσιο διαλόγου λειτουργίας εκτός σύνδεσηςεφαρμογές που εκτελούνται από το Navigator.
- Τροποποιήσουμε την οθόνη του Navigator με την επιλογή Enable High DPI scaling. Αυτή η επιλογή μπορεί να είναι χρήσιμη εάν το Navigator δεν εμφανίζεται σωστά σε ορισμένες οθόνες υψηλού DPI.
- Εμφανίσουμε μηνύματα σφάλματος εκκίνησης εφαρμογής.
- Ορίσουμε τη διαδρομή PyCharm Community Edition εάν δεν ήταν εγκατεστημένη στη προεπιλεγμένη θέση.
- Ορίσουμε τη διαδρομή PyCharm Pro εάν δεν ήταν εγκατεστημένη στη προεπιλεγμένη θέση.
- Ορίσουμε τη διαδρομή κώδικα του Visual Studio εάν δεν ήταν εγκατεστημένη στη προεπιλεγμένη θέση.

Υπόδειξη

Κάντε κλικ στο κουμπί **Reset to defaults** (Επαναφορά στις προεπιλογές) για να αλλάξετε τις προτιμήσεις στις προεπιλεγμένες τους τιμές.

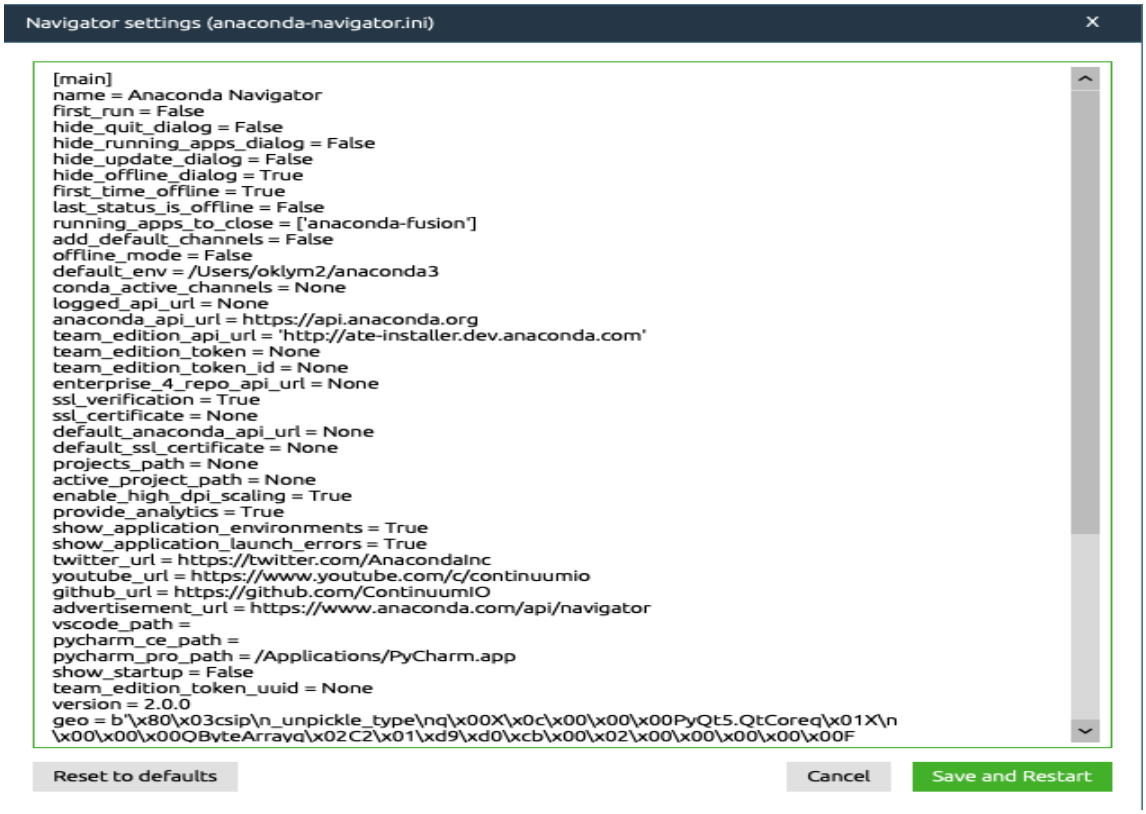




Προσοχή

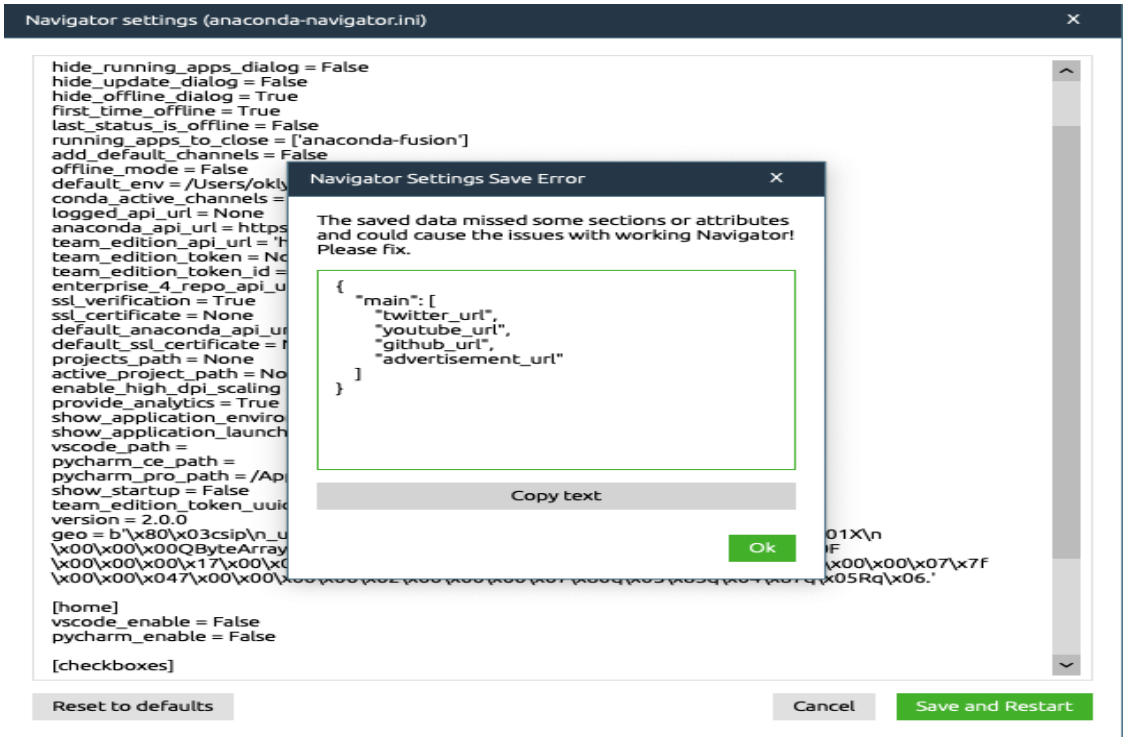
Να είστε προσεκτικοί όταν αλλάζετε τιμές απευθείας στα αρχεία διαμόρφωσης για το Navigator ή το conda. Η εσφαλμένη διαμόρφωση μπορεί να προκαλέσει προβλήματα με αυτά τα προϊόντα.

Πραγματοποιήστε αλλαγές στο αρχείο διαμόρφωσης του Navigator (anaconda-navigator.ini).

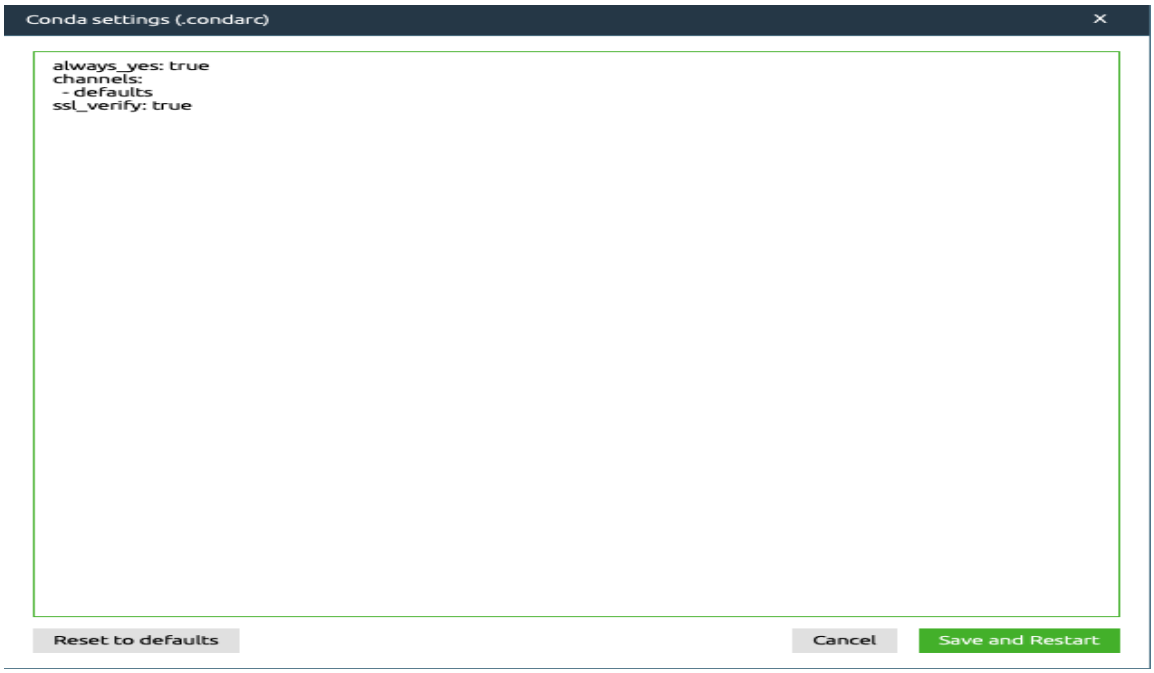


Σημείωση

Περιλαμβάνεται ένα εργαλείο επικύρωσης βασικής διαμόρφωσης, επομένως εάν ο χρήστης χάσει οποιαδήποτε απαιτούμενη ενότητα ή επιλογή θα ενημερωθεί μέσω ενός αναδυόμενου πλαισίου. Δείτε το παρακάτω σχήμα.

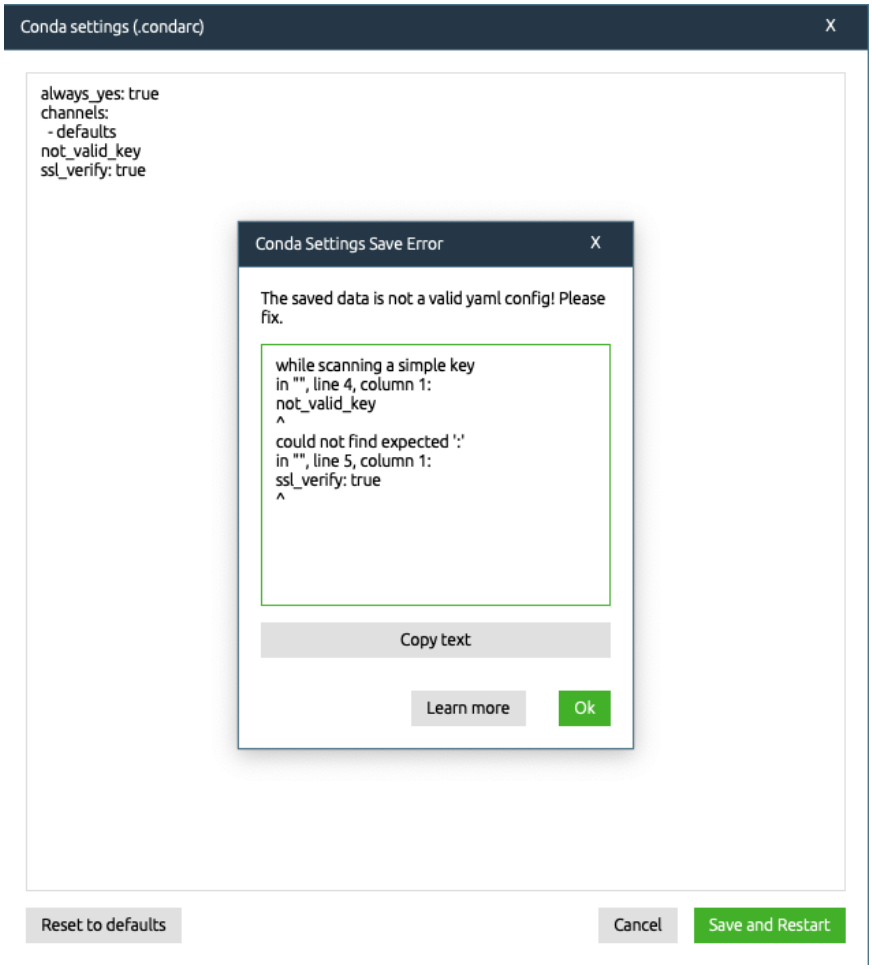


Κάνουμε αλλαγές στο αρχείο διαμόρφωσης Conda (.condarc).



Σημείωση

Περιλαμβάνεται ένα εργαλείο επικύρωσης βασικής διαμόρφωσης. Το αρχείο .condarc πρέπει να είναι έγκυρο αρχείο μορφής yaml. Εάν το περιεχόμενο δεν είναι σε έγκυρη μορφή yaml, ο χρήστης θα λάβει ένα σφάλμα. Δείτε το παρακάτω σχήμα.



- Υπηρεσίες (μόνο για macOS) – σύνδεσμοι προς το μενού προτιμήσεων συστήματος του υπολογιστή σας.
- Απόκρυψη Anaconda-Navigator (μόνο για macOS) – αποκρύπτει το παράθυρο του Navigator.
- Απόκρυψη άλλων (μόνο για macOS) – αποκρύπτει όλα τα παράθυρα εκτός από το παράθυρο του Navigator.
- Εμφάνιση όλων (μόνο για macOS) – εμφανίζει όλα τα παράθυρα συμπεριλαμβανομένου του παραθύρου του Navigator.

- Κλείστε το Anaconda-Navigator–έξοδοι από το Navigator.

Μενού Help

Το μενού **Help** περιέχει τις ακόλουθες επιλογές:

- Αναζήτηση : σύνδεσμοι στη Βοήθεια του υπολογιστή σας (μόνο για Windows και macOS).
- Ηλεκτρονική Τεκμηρίωση : σύνδεσμοι προς αυτήν τη τεκμηρίωση, την οποία μπορούμε να διαβάσουμε σε οποιοδήποτε πρόγραμμα περιήγησης ιστού. Μπορούμε επίσης να ανοίξουμε τη τεκμηρίωση κάνοντας κλικ στο κουμπί Documentation στο κάτω αριστερό μέρος του παραθύρου του Navigator.
- Προβολή αρχείων καταγραφής : μας επιτρέπει να ελέγξουμε τα αρχεία καταγραφής όλων των ενεργειών που πραγματοποιήθηκαν στο Navigator στην τρέχουσα περίοδο λειτουργίας. Αυτή η επιλογή εμφανίζει μία λίστα αρχείων καταγραφής, συμπεριλαμβανομένου navigator.log, που περιέχει αρχεία καταγραφής εφαρμογών Navigator και condamanager.log, που περιέχει αρχεία καταγραφής γραμμένα από το στοιχείο conda-manager.

Σημείωση

Ένα νέο αρχείο καταγραφής δημιουργείται κάθε φορά που εκτελείτε το Navigator, με έναν διαδοχικό αριθμό προσαρτημένο στο όνομα του αρχείου. Τα πιο πρόσφατα αρχεία καταγραφής έχουν υψηλότερους αριθμούς.

Κουμπιά Navigator window

- Συνδεθείτε στο anaconda.org–εμφανίζεται επάνω δεξιά. Κάντε κλικ για να συνδεθείτε στο anaconda.org και να ενεργοποιήσετε την αναζήτηση πακέτων σε αυτό. Αφού συνδεθείτε, η ετικέτα του κουμπιού αλλάζει σε "Σύνδεση ως [όνομα χρήστη]".
- Τεκμηρίωση – εμφανίζεται κάτω αριστερά. Κάντε κλικ για να ανοίξετε την τεκμηρίωση του Navigator σε ένα πρόγραμμα περιήγησης.
- Ιστολόγιο προγραμματιστή–εμφανίζεται κάτω αριστερά. Κάντε κλικ για να διαβάσετε τι λένε οι προγραμματιστές σχετικά με την ανάπτυξη του Navigator.
- Μέσα κοινωνικής δικτύωσης – εμφανίζεται κάτω αριστερά. Κάντε κλικ για να δείτε τις σελίδες του Anaconda στο Twitter, στο YouTube και στο GitHub.

Συμπεράσματα

Η τεχνητή νοημοσύνη (ΤΝ) είναι πλέον μέρος του καθημερινού μας περιβάλλοντος, είτε το γνωρίζουμε είτε όχι. Στη νέα εποχή, οι άνθρωποι βλέπουν την ευκαιρία να απαλλαγούν από την καταναγκαστική, μη δημιουργική εργασία, αντικαθιστώντας εαυτούς, με μηχανές. Κάποια βασικά στοιχεία αυτής της επανάστασης των μηχανών, παρουσιάστηκαν σε αυτό το πόνημα. Η προσπάθεια κλωνοποίησης της ανθρώπινης νοημοσύνης στις μηχανές, έχει ενισχυθεί με την εφαρμογή στην πράξη των συστημάτων ασαφούς λογικής που αναπτύχθηκαν στην εργασία. Έγινε φανερό πως από την χαρτογράφηση του εγκεφάλου, οι επιστήμονες δημιούργησαν μαθηματικό αντίγραφο της δομής και λειτουργείας των φυσικών νευρώνων. Προχώρησαν στην κατασκευή αλγορίθμων του τρόπου σκέψης, της ανάκλησης μνήμης και της ποσοτικής ανάλυσης, που διενεργεί ο ανθρώπινος εγκέφαλος. Οι αλγόριθμοι εξελίχθηκαν σε μοντέλα, κάποια εξ΄αυτών (CNN, RNN-LSTM) υλοποιήθηκαν στην εργασία. Η γνώση αυτή, ενσωματώθηκε όπως είδαμε σε βιβλιοθήκες λογισμικού (TensorFlow, Keras κα) για να γίνει εφικτή η επικοινωνία με τις μηχανές. Έγινε προσπάθεια να δοθεί μια μικρή εικόνα για το τεχνικό υπόβαθρο (θεωρία, λογισμικό), που κρύβεται στο παρασκήνιο όταν για παράδειγμα ανοίγουμε το Facebook ή το Twitter, κάνουμε μία αναζήτηση στο Google, αγοράζουμε μία πρόταση από την Amazon ή κάνουμε κράτηση για ένα ταξίδι στο διαδίκτυο. Σήμερα έχουμε ανάγκη την τεχνητή νοημοσύνη ακόμα και στο καθημερινό μας ταξίδι από και προς τη δουλειά. Οι εφαρμογές πλοήγησης όπως οι χάρτες Google χρησιμοποιούν τεχνητή νοημοσύνη για να αξιολογήσουν την ταχύτητα κίνησης της κυκλοφορίας. Έχει επιρροή στις τραπεζικές δραστηριότητες, από τον εντοπισμό απάτης μέχρι την εξυπηρέτηση πελατών και τις επενδύσεις. Τα chatbots αναγνωρίζουν λέξεις και φράσεις προκειμένου να παρέχουν σχετικές πληροφορίες στους πελάτες με τυπικές ερωτήσεις. Τα chatbots μπορεί να είναι τόσο ακριβή που φαίνεται σαν να συνομιλούμε με ένα πραγματικό άτομο. Προσπαθούν να μιμηθούν τη φυσική γλώσσα μιμούμενοι συνομιλίες ενώ βοηθούν σε καθημερινές εργασίες όπως προγραμματισμός ραντεβού, αποδοχή παραγγελιών και απάντηση σε ερωτήσεις χρέωσης. Οι βοηθοί φωνής, όπως ο βοηθός Google, η Alexa και η Siri, είναι τα καλύτερα παραδείγματα τεχνητής νοημοσύνης στην πραγματική ζωή. κάντε ερώτηση μέσω φωνής, επεξεργαστείτε την με τις τεχνολογίες αναγνώρισης ομιλίας και επεξεργασίας φυσικής γλώσσας του τηλεφώνου και στη συνέχεια, παραδώστε τα αποτελέσματα ως ομιλία ή κείμενο. Τα drones ή μη επανδρωμένα εναέρια οχήματα (UAV), είναι ήδη παρόντα στους ουρανούς μας, πραγματοποιούν επιτήρηση και παρέχουν υπηρεσίες παράδοσης, μεταξύ των οποίων η παράδοση φαρμάκων και ειδών πρώτης ανάγκης σε περιορισμένους στο σπίτι ηλικιωμένους

από Covid-19. Ο εντοπισμός και η διάγνωση ασθενειών έχει γίνει λίγο πιο απλός χάρη στην ανάπτυξη ρομπότ που λειτουργούν με τεχνητή νοημοσύνη. Επιπλέον, συμβάλλει σημαντικά κάνοντας τις διαδικασίες θεραπείας και διαχείρισης πιο απλοποιημένες. Με τις τεχνολογικές προόδους, όπως η αναγνώριση αντικειμένων και η αναγνώριση προσώπου, δεν θα αργήσει η παρακολούθηση όλων σχεδόν των καμερών ασφαλείας από τεχνητή νοημοσύνη και όχι από ανθρώπους.

Η βάση για την ανάπτυξη και εξέλιξη όλων των συστημάτων υπολογιστικής νοημοσύνης, είναι η συνεχή τροφοδοσία τους με μεγάλους όγκους δεδομένων. Ο όγκος δεδομένων που παράγεται στον κόσμο αναμένεται να αυξηθεί από 33 zettabytes το 2018 σε 175 το 2025 (1 zettabyte είναι ένα τρισεκατομμύριο gigabytes). Έτσι χάρη στη συνεχή αύξηση των δεδομένων που είναι διαθέσιμα στους αλγόριθμους μηχανικής μάθησης, η ΤΝ εξελίσσεται. Αυτό επιτρέπει να λειτουργεί πιο αποτελεσματικά, επειδή τα εκπαιδευμένα μοντέλα ML μπορούν να διαμορφωθούν ώστε να λειτουργούν καλύτερα με το ευρύ φάσμα πραγματικών δεδομένων των χρηστών. Τα παραπάνω είναι μερικά από τα πιο χαρακτηριστικά παραδείγματα ΤΝ στην καθημερινή ζωή που είναι ευρέως διαδεδομένα και εύχρηστα. Αυτό αποδεικνύει ότι η ΤΝ αλλάζει ήδη τη ζωή μας, επιτρέποντάς μας να είμαστε πιο παραγωγικοί, ενώ παράλληλα καταβάλλουμε τις προσπάθειές μας σε πραγματικά προβλήματα. Αυτή η τεχνολογία θα επιταχύνει, θα επεκταθεί και θα γίνει πιο σημαντική για όλες τις βιομηχανίες και σχεδόν κάθε πτυχή της καθημερινής μας ζωής στο μέλλον.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. *Krizhevsky Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.*
2. *Liu Wei, et al. "SSD: Single Shot MultiBox Detector." European conference on computer vision. Springer, Cham, 2016.*
3. *Girshick Ross. "Fast R-CNN." Proceedings of the IEEE international conference on computer vision. 2015.*
4. *Lin Tsung-Yi, et al. "Focal loss for Dense Object Detection. "Proceedings of the IEEE international conference on computer vision. 2017.*
5. *Bodla Navaneeth, et al. "Soft-NMS--Improving Object Detection With One Line of Code." Proceedings of the IEEE international conference on computer vision. 2017.*
6. *Pramod Singh, Avinash Manure, "Learn TensorFlow 2.0 Implement Machine Learning and Deep Learning Models with Python". 2020*
7. *Nishhant Shukla with Kenneth Fricklas, "Machine Learning with TensorFlow". 2020.*
8. *Bernard Mar with Matt Ward, "ARTIFICIAL INTELLIGENCE IN PRACTICE". 2018.*
9. *Sam Abrahams, Danijar Hafner, Erik Erwitt, Ariel Scarpinelli, "TensorFlow for Machine Intelligence". 2016.*
10. *Zhicai Liu, Jose Carlos R. A;cantud, Keyun Qin, and Ling Xiong, "The Soft Sets and Fuzzy Sets-Based Neural". 2021.*
11. *Fernando Gaxiola, Patricia Melin, Fevrier Valdez, "New Backpropagation,Algorithm with Type-2 Fuzzy Weights for Neural Networks". 2016.*
12. *Anaconda , https://www.anaconda.com/*
13. *Google Colaboratoryoratory, https://colaboratory.research.google.com/*
14. *Tensorflow , https://www.tensorflow.org/*