Πανεπιστήμιο Πειραιώς

Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης
Πρόγραμμα Μεταπτυχιακών Σπουδών
Εφαρμοσμένη Στατιστική

Πρόβλεψη Τροχιών σε Δεδομένα Κίνησης με Βαθιά Νευρωνικά Δίκτυα

Παπαδόπουλος Ν. Αθανάσιος
ΜΕΣ 16001

Διπλωματική Εργασία

που υποβλήθηκε στο τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς ως μέρος των απαιτήσεων για την απόκτηση Μεταπτυχιακού Διπλώματος Ειδίκευσης στην Εφαρμοσμένη Στατιστική

Πειραιάς,
Δεκέμβριος 2018

Πανεπιστήμιο Πειραιώς

Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης
Πρόγραμμα Μεταπτυχιακών Σπουδών
Εφαρμοσμένη Στατιστική

Πρόβλεψη Τροχιών σε Δεδομένα Κίνησης με Βαθιά Νευρωνικά Δίκτυα

Παπαδόπουλος Ν. Αθανάσιος
ΜΕΣ 16001

Διπλωματική Εργασία

που υποβλήθηκε στο τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς ως μέρος των απαιτήσεων για την απόκτηση Μεταπτυχιακού Διπλώματος Ειδίκευσης στην Εφαρμοσμένη Στατιστική

Πειραιάς,
Δεκέμβριος 2018

Η παρούσα Διπλωματική Εργασία εγκρίθηκε ομόφωνα από την Τριμελή Εξεταστική Επιτροπή που ορίσθηκε από τη Γ.Σ.Ε.Σ του τμήματος Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς στην υπ' αριθμό ..... συνεδρίασή του σύμφωνα με τον Εσωτερικό Κανονισμό Λειτουργίας του Προγράμματος Μεταπτυχιακών Σπουδών στην Εφαρμοσμένη Στατιστική.

Τα μέλη της Επιτροπής ήταν:

- Επίκουρος Καθηγητής, Πελέκης Νικόλαος (Επιβλέπων)
- Καθηγητής, Θεοδωρίδης Ιωάννης
- Αναπληρωτής Καθηγητής, Κοφίδης Ελευθέριος

Η έγκριση της Διπλωματικής Εργασίας από το τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς δεν υποδηλώνει αποδοχή των γνωμών του συγγραφέα.

University of Piraeus

Department of Statistics and Insurance Science
Postgraduate Program In
Applied Statistics

Trajectory Prediction in Mobility Data with Deep Neural Networks

Papadopoulos N. Athanasios
MES 16001

MSc Dissertation

submitted to the Department of Statistics and Insurance Science of the University of Piraeus in partial fulfillment of the requirements for the degree of Master of Science in Applied Statistics.

Piraeus, Greece
December 2018

This thesis was approved unanimously be the three – member committee appointed by the Department of Statistics and Actuarial Science, University of Piraeus, in accordance with the rules of the MSc program in Applied Statistics.

Committee members were:

- Assistant Professor, Pelekis Nikolaos (Supervisor)
- Professor, Theodoridis Ioannis
- Associate Professor, Kofidis Eleftherios

Approval of this thesis from the Department of Statistics and Actuarial Science, University of Piraeus, does not imply any endorsement of the opinions of the author.

Στην Οικογένειά μου
και στην κοπέλα μου,

Ευχαριστίες

Θα ξεκινήσω ευχαριστώντας την οικογένεια μου και την κοπέλα μου, που σε αυτή την προσπάθεια, ήταν καθ' όλη τη διάρκεια δίπλα μου και με στήριζαν. Φυσικά, δεν πρέπει να ξεχάσω τους στενούς μου φίλους, που βρίσκονται συνεχώς στο πλευρό μου.

Επίσης, δεν θα μπορούσα να μην ευχαριστήσω τον καθηγητή μου κ. Νίκο Πελέκη, που μου έδωσε την δυνατότητα να ασχοληθώ με ένα τέτοιο θέμα, καθώς και για τις συμβουλές και τις ουσιαστικές παρατηρήσεις που με βοήθησαν στην διεκπεραίωση αυτή της εργασίας. Ενώ, θα ήθελα να τον ευχαριστήσω και για την κατανόηση που έδειξε στην πορεία της εργασίας.

Τέλος, θα ήθελα να αναφερθώ στο σύνολο των καθηγητών του μεταπτυχιακού προγράμματος Εφαρμοσμένης Στατιστική, για τις γνώσεις που μου προσέφεραν.

Αθανάσιος Παπαδόπουλος

Περίληψη

Σκοπός της παρούσας εργασίας αποτελεί η μελέτη της χρήσης νευρωνικών δικτύων και έπειτα η εφαρμογή τους σε ένα σετ δεδομένων κίνησης, το οποίο αποτελείται από τις τροχιές αεροσκαφών από τη Μαδρίτη στην Βαρκελώνη κατά την περίοδο του Απριλίου του 2016.

Το πρόβλημα της δημιουργίας προβλέψεων τροχιών σε κινούμενα δεδομένα, είναι ένα ολοένα και περισσότερο περιζήτητο φαινόμενο, αφού αυξάνεται συνεχώς το μέγεθος των δεδομένων που συλλέγονται από συστήματα καταγραφής της θέσης *(GPS)*. Αυτό αποτελεί μία πρόκληση για την επιστημονική κοινότητα, η οποία προσπαθεί μέσω αυτών των δεδομένων, να κατανοήσει πως εξελίσσονται οι τροχιές τόσο σε βραχυπρόθεσμο, όσο και σε μακροπρόθεσμο επίπεδο, ώστε να είναι ικανοί να προβλέψουν ανωμαλίες και όχι μόνο, στην πορεία των αντικειμένων που μελετάνε.

Τα νευρωνικά δίκτυα με τα οποία θα ασχοληθούμε, αποτελούν μέθοδο Βαθιάς Μάθησης και είναι μέρος της Τεχνητής Νοημοσύνης. Είναι υπολογιστικά συστήματα εμπνευσμένα από τα βιολογικά νευρωνικά δίκτυα του ανθρώπινου εγκεφάλου. Έτσι ο αρχικός σκοπός τους ήταν να δώσουν λύσεις σε προβλήματα, με τον ίδιο τρόπο που δίνει λύσεις ο ανθρώπινος εγκέφαλος, δηλαδή μέσω της εμπειρίας που λαμβάνουν από παρελθοντικά γεγονότα. Έχουν δημιουργηθεί αρκετές παραλλαγές τους, εκ των οποίων η κάθε μία δίνει λύσεις σε διαφορετικά προβλήματα. Στη συγκεκριμένη εργασία, θα ασχοληθούμε εκτενέστερα με το είδος των επαναλαμβανόμενων νευρωνικών δικτύων.

Ξεκινάμε την εργασία, παρουσιάζοντας κάποια εισαγωγικά στοιχεία, όσον αφορά την έννοια της τροχιάς και της κίνησης αντικειμένων στον χώρο και στο χρόνο. Στο Κεφάλαιο 2, αναφερόμαστε στην έννοια την Βαθιάς Μάθησης και στα χαρακτηριστικά της, με σκοπό να προχωρήσουμε στο Κεφάλαιο 3 στην εισαγωγή των Νευρωνικών Δικτύων και στο πως αυτά λειτουργούν και εκπαιδεύονται. Στο Κεφάλαιο 4, εισερχόμαστε στο κομμάτι των Επαναλαμβανόμενων Νευρωνικών Δικτύων, που είναι και το κύριο κομμάτι ενδιαφέροντος της παρούσας εργασίας, αφού αυτά θα χρησιμοποιήσουμε για την εφαρμογή των μοντέλων μας. Πριν προχωρήσουμε όμως σε αυτή, παρουσιάζουμε ορισμένες ενδιαφέρουσες μεθόδους υλοποίησης παρόμοιων προβλημάτων πρόβλεψης τροχιών. Στο τελευταίο κεφάλαιο, παρουσιάζουμε την υλοποίηση της εργασίας μας, όπου χρησιμοποιούμε τα επαναλαμβανόμενα νευρωνικά δίκτυα προκειμένου να προβλέψουμε την μελλοντική θέση του αεροσκάφους, βασιζόμενο στην αμέσως προηγούμενη θέση του.

Abstract

The purpose of this thesis is to study the use of neural networks and then to apply them to a mobility data set, which consists of the aircrafts' trajectories from Madrid to Barcelona during April of 2016.

The problem of analytics prediction in mobility data is an extremely popular phenomenon, as the amount of mobility data that are generated from Global Positioning Systems *(GPS)* is constantly increasing. This is a challenging task for the scientific community, which is trying through these data, to understand how the moving entities are evolving both in the short and the long term, so they can predict abnormalities and more in the trajectory of the objects they are studying.

The neural networks that we will deal with are a method of Deep Learning and part of Artificial Intelligence. They are computer systems inspired by the biological neural networks of the human brain. So, their original purpose was to provide solutions to problems, in the same way the human brain does, that is through the experience they receive from past events. Several variants have been created, each of which gives solutions to different problems. In this paper, we will deal more closely with the type of recurrent neural networks.

We begin our work, presenting some introductory elements, regarding the meaning of trajectory and the movement of objects in space and time. In Chapter 2, we refer to the concept of Deep Learning and its characteristics, in order to move to Chapter 3 of introducing Neural Networks and how they are working and how they are training. In Chapter 4, we enter the part of the Recurrent Neural Networks, which is the main piece of interest in the thesis, since we will use them to implement our model. Before proceeding with this, we present some interesting methods of similar trajectory problems. In the last chapter, we present the implementation of our work, where we use LSTM neural networks to predict the future position of the aircraft, based on its previous position in space and time.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1

Τροχιές Κινούμενων Αντικειμένων

Ο κόσμος γύρω μας κατακλύζεται από κίνηση κάθε είδους και μορφής, από τα έμψυχα όντα όπως είναι ο άνθρωπος και τα ζώα, έως τα άψυχα όπως είναι τα αυτοκίνητα και άλλες δημιουργίες του ανθρώπου στο χρόνο. Η μελέτη και η καταγραφή των κινήσεων αυτών καθώς και ο τρόπος που αλληλοεπιδρούν ως προς το εξωτερικό τους περιβάλλον αλλά και μεταξύ τους τα διάφορα αντικείμενα, αποτελεί ένα πολύ ενδιαφέρον πεδίο μελέτης από διάφορες επιστήμες, καθώς μέσα από αυτά μπορεί να εξαχθούν πολύ χρήσιμα συμπεράσματα για την συμπεριφορά των αντικειμένων έμψυχων ή άψυχων που μας περιβάλλουν, καθώς και για το κατά πόσο η συμπεριφορά τους αυτή μας επηρεάζει άμεσα ή έμμεσα. Στο κεφάλαιο αυτό θα παρουσιάσουμε συνοπτικά την έννοια της κίνησης και των τροχιών των κινούμενων αντικειμένων, το οποίο μας απασχολεί και στην παρούσα εργασία, ξεκινώντας από μία σύντομη εισαγωγή για τις έννοιες του χώρου και του χρόνου.

1.1 Χώρος και Χρόνος

Οι έννοιες του χώρου και του χρόνου είναι τόσο «προφανείς», αλλά παράλληλα, συνιστούν και ένα μέγα μυστήριο. Ο χρόνος και ο χώρος είναι οι δυο θεμελιώδεις έννοιες από τις οποίες αρχίζει η πορεία αναζήτησης της γνώσης και ερμηνείας της φυσικής πραγματικότητας, σύμφωνα με τις οποίες οι ζωές μας εξελίσσονται. Ως χώρος γενικά στη φυσική ονομάζεται το περιβάλλον που είναι άρρηκτα συνδεδεμένο στην ιδέα της απόστασης σε τρεις διαστάσεις. Ο χρόνος νοείται ως μια ανεξάρτητη γραμμή, κάτι σαν σιδηροδρομική γραμμή, που εκτείνεται επ'άπειρο προς τις δυο κατευθύνσεις και θεωρείται παντοτινός υπό την έννοια ότι είχε υπάρξει από πάντα και θα υπάρχει για πάντα, σύμφωνα με τον Stephen Hawking στο βιβλίο του "Το Χρονικό του Χρόνου". Τα πάντα όμως, κινούνται όχι μόνο στο χώρο αλλά και στον χρόνο, ο χρόνος θεωρείται τέταρτη διάσταση του χώρου. Έτσι οι τέσσερις πλέον διαστάσεις λέγονται χωροχρόνος ή χωροχρονικό συνεχές. Το πρώτο μαθηματικό πρότυπο για το χρόνο και το χώρο, το έδωσε ο Νεύτωνας, στο σύγγραμμά του "Μαθηματικές Αρχές της Φυσικής Φιλοσοφίας". Στο πρότυπο αυτό, ο χρόνος και ο χώρος είναι διαχωρισμένοι μεταξύ τους και συνιστούν ένα υπόβαθρο επάνω στο οποίο διαδραματίζονται τα γεγονότα χωρίς όμως να το επηρεάζουν. Ένα γεγονός είναι κάτι που συμβαίνει σε κάποια συγκεκριμένη στιγμή στο χρόνο και σε κάποιο συγκεκριμένο σημείο στο χώρο και συνήθως ερμηνεύεται μέσα από τέσσερις διαστάσεις, ως συνδυασμός του ευκλείδειου χώρου τριών διαστάσεων και του χρόνου ($R^4 = R^3 \times R$) με παραμέτρους (x, y, z, t) (Πνευματικός, 2006)

1.2 Κινούμενα Αντικείμενα

Η κίνηση, αποτελεί βασικό χαρακτηριστικό πολλών δραστηριοτήτων και διαδικασιών και η κατανόηση της είναι πολύ βασική για πολλούς τομείς της επιστήμης. Για αυτό το λόγο, όλο και συχνότερα συλλέγονται πλέον δεδομένα παρακολούθησης κίνησης διάφορων αντικειμένων, τα οποία είναι συνήθως πολύ μεγάλα σε όγκο και σύνθεση. Έτσι πολύ αναλυτές πλέον ασχολούνται με τον τομέα αυτό και την προσπάθεια ανίχνευσης μοτίβων κίνησης σε κινούμενα αντικείμενα.

Ως κινούμενο αντικείμενο, νοείται κάθε αντικείμενο το οποίο παρέχει πληροφορίες και δεδομένα της κίνησής του, προερχόμενα από διάφορες πηγές, όπως για παράδειγμα τα δεδομένα που καταγράφονται από συσκευές GPS ή τηλεφωνικές συσκευές. Κάθε κινούμενο αντικείμενο, έχει την δική του τοποθεσία στον χώρο, η οποία μπορεί να μεταβάλλεται με το πέρασμα του χρόνου. Η κίνηση που εκτελεί εξαρτάται άμεσα από τη φύση του. Κάποια αντικείμενα μπορεί να εκτελούν προκαθορισμένη κίνηση, όπως τα δρομολόγια αεροπλάνων, ενώ κάποια άλλα ακανόνιστη κίνηση, όπως τα ζώα που περιπλανιούνται στο δάσος. Επίσης, η κίνηση του κάθε αντικειμένου, εξαρτάται άμεσα από τον χώρο αλλά και τα λοιπά αντικείμενα που τον περιβάλλουν, όπως οι πορείες των κομητών και των πλανητών στο χώρο του διαστήματος, εξαρτώνται από τις βαρυτικές έλξεις των γύρω σωμάτων του χώρου στον οποίο κινούνται. Η κινητικότητα αυτή των αντικειμένων, δημιουργεί κίνηση *(traffic),* από την οποία δημιουργούνται μοτίβα *(patterns).* (Χονδροδήμα, 2013)

1.2.1 Δεδομένα Κίνησης

Οι καθημερινές δραστηριότητες, ο τρόπος με τον οποίο ζούμε και κινούμαστε, αφήνει πίσω του ί-χνη*(traces),* τα οποία με τις σύγχρονες τεχνολογίες, δύναται αυτά να καταγραφούν και να εξαχθούν χρήσιμα συμπεράσματα για τον τρόπο με τον οποίο συμπεριφερόμαστε στην καθημερινότητα μας εμείς, αλλά και αντικείμενα ενδιαφέροντος σε διάφορες εφαρμογές της καθημερινότητας αυτής.

Από αρχαιοτάτων χρόνων, ο άνθρωπος ξεκίνησε να παρατηρεί τις συμπεριφορές διάφορων κινούμενων αντικειμένων, από τις κινήσεις των ζώων μέχρι των αστεριών. Αν και οι μέθοδοι που χρησιμοποιήθηκαν σε παλαιότερες εποχές για την παρατήρηση, τη μέτρηση, την καταγραφή και την ανάλυση των κινήσεων είναι πολύ διαφορετικές από αυτές που χρησιμοποιούνται σήμερα με τις σύγχρονες τεχνολογίες, υπάρχουν ακόμη πολλά που πρέπει να μάθουμε από τις μελέτες του παρελθόντος. Αρχικά, η πλήρης προσοχή που δίνεται στις διάφορες πτυχές της κίνησης, πέραν της τροχιάς του αντικειμένου στο χώρο, αλλά και ορισμένα βασικά χαρακτηριστικά αυτής, όπως η ταχύτητα και η κατεύθυνση. Επίσης, η μελέτη της συσχέτισης της κίνησης με τις ιδιότητες του περιβάλλοντός στο οποίο αυτή διαπράττεται και με τα διάφορα φαινόμενα που το διέπουν.

Τα δεδομένα κίνησης, αποτελούν ένα πλήθος πληροφοριών τα οποία μπορούν να προέρχονται από διάφορες πηγές, όπως τηλεφωνικές συνομιλίες, δεδομένα από συστήματα GPS, δεδομένα από σημεία πρόσβασης Wi-fi και λοιπές πηγές. Όμως, ανάλογα και το είδος της πηγής των δεδομένων, αυτά μπορούν να διαφέρουν τόσο σε μέγεθος, όσο και σε ανάλυση. Το πιο διαδεδομένο είδος πηγής τέτοιων δεδομένων, λόγω της υψηλής ακρίβειας τους, αποτελούν τα συστήματα GPS.

Έτσι, μας παρέχεται σήμερα η δυνατότητα συλλογής και αποθήκευσης δεδομένων κίνησης σε ποσότητα αλλά και ποιότητα που δεν μπορούσαμε στο παρελθόν να έχουμε και μάλιστα σε σχετικά πολύ χαμηλό κόστος. Παρόλα αυτά όμως, υπάρχει μεγάλος δρόμος από τα δεδομένα κίνησης στο να καταφέρουμε να τα μετατρέψουμε σε γνώση *(mobility knowledge)* ικανή να εξάγουμε χρήσιμα συμπεράσματα.

1.2.2 Χαρακτηριστικά Δεδομένων Κίνησης

Το πιο γνωστό χαρακτηριστικό των δεδομένων κίνησης και γενικότερα των χωρικών δεδομένων, αφορά την χωρική αυτοσυσχέτιση *(spatial autocorrelation)* και πηγάζει από τον πρώτο νόμο της επιστήμης της γεωγραφίας. Αυτός αναφέρεται στο ότι, *«Το καθετί είναι συσχετιζόμενο με οτιδήποτε άλλο γύρω του, αλλά τα κοντινότερα αντικείμενα, τείνουν να είναι πιο συσχετιζόμενα σε σχέση με τα μακρινότερα σε απόσταση από αυτά».* Η χωρική αυτή εξάρτηση των αντικειμένων, μας δείχνει ότι το πλαίσιο στο οποίο μελετάμε τα διάφορα αντικείμενα, έχει ιδιαίτερο αντίκτυπο στη διαδικασία αυτή. Αυτό, έρχεται σε

αντίθεση με την κλασσική στατιστική θεωρία, που κάνει την υπόθεση ότι οι παρατηρήσεις είναι ανεξάρτητες μεταξύ τους, αλλά αυτές ακολουθούν πανομοιότυπες κατανομές*(i.i.d)*.

Ένα δεύτερο βασικό χαρακτηριστικό των χωρικών δεδομένων, είναι η ετεροσκεδαστικότητα. Αυτή προκύπτει από το γεγονός ότι τα χωρικά δεδομένα, σπανίως παρουσιάζουν σταθερά χαρακτηριστικά. Η χωρική ετεροσκεδαστικότητα σχετίζεται με την έλλειψη σταθερότητας στη συμπεριφορά των σχέσεων στο χώρο. Αυτό το χαρακτηριστικό είναι επίσης γνωστό ως μη στασιμότητα*(nonstationarity)*.

Για περισσότερες πληροφορίες, ο αναγνώστης μπορεί να μελετήσει το paper των (Bacao, Lobo, & Painho, 2005), στο οποίο καταγράφονται πιο λεπτομερή τα χαρακτηριστικά αυτά.

Όσον αφορά τα δεδομένα κίνησης συγκεκριμένα, για την αναγνώριση της δραστηριότητας ενός κινούμενου αντικειμένου, πρέπει να ληφθούν υπόψιν αρκετοί παράγοντες που σχετίζονται με τα χαρακτηριστικά και τη φύση αυτής. Όσον αφορά τα χαρακτηριστικά της κίνησης, αυτά μπορούν να ταξινομηθούν σε δύο κατηγορίες. (Τραγοπούλου, 2013)

Στην πρώτη κατηγορία αναφερόμαστε στα χαρακτηριστικά της κίνησης που μπορούν να καταγραφούν σε μία συγκεκριμένη χρονική στιγμή και αυτά είναι:

- Χρονοσφραγίδα*(timestamp)*
- Θέση στο χώρο
- Κατεύθυνση
- Ταχύτητα
- Αλλαγή Κατεύθυνσης
- Επιτάχυνση
- Συνολικός Χρόνος Κίνησης
- Συνολική Απόσταση Κίνησης

Στην δεύτερη κατηγορία, αναφερόμαστε στα χαρακτηριστικά που περιγράφουν την κίνηση σε ένα χρονικό διάστημα και αυτά είναι:

- Γεωμετρικό σχήμα κίνησης στο χώρο
- Απόσταση στο χώρο
- Χρονική διάρκεια κίνησης
- Διάνυσμα της κίνησης *(από την αρχική στην τελική θέση)*
- Μέση και μέγιστη ταχύτητα
- Δυναμική συμπεριφορά ταχύτητας *(επιτάχυνση, επιβράδυνση, μηδενική, σταθερή ταχύτητα)*
- Δυναμική συμπεριφορά κατεύθυνσης

1.3 Τροχιές Κινούμενων Αντικειμένων

Η ιδέα της τροχιάς προέρχεται από την ικανότητα να καταγραφεί *(capture)* η κίνηση ενός αντικειμένου που κινείται στο γεωγραφικό χώρο για κάποιο χρονικό διάστημα. (Τραγοπούλου, 2013)

Η καταγραφή της κίνησης γίνεται για κάθε κινούμενο αντικείμενο και για κάθε διαδρομή κίνησής του και μπορεί να γίνει με διάφορους τρόπους, ανάλογα με τον τρόπο παρατήρησης της:

- Χρονική. Η καταγραφή της θέσης, γίνεται ανά τακτά χρονικά διαστήματα.

- Με βάση την μεταβολή της θέσης. Γίνεται καταγραφή, όταν η θέση του αντικειμένου, διαφέρει από την προηγούμενη.
- Με βάση τη θέση κοντά σε συγκεκριμένη τοποθεσία.
- Με βάση κάποιο συμβάν.

Μία διαδρομή κίνησης *(movement track)* αποτελείται βασικά από μία χρονική ακολουθία χωροχρονικών θέσεων, δηλαδή ζεύγη *(χρονικό στίγμα, σημείο*), που καταγράφονται για το κινούμενο αντικείμενο. Ωστόσο, ανάλογα με τις δυνατότητες της συσκευής, συμπληρωματικά στοιχεία, π.χ. η στιγμιαία ταχύτητα ή ακινησία, η επιτάχυνση, η κατεύθυνση και η περιστροφή, μπορούν να συμπληρώνουν τα ζεύγη. Τα δεδομένα που καταγράφονται από τη συσκευή αποκαλούνται ακατέργαστα δεδομένα *(raw data)*.

Από την προοπτική μοντελοποίησης, στον γεωγραφικό χώρο, μπορούμε να αναγνωρίσουμε ένα σύνολο από βασικές χωρικές οντότητες, σημεία, γραμμές και επιφάνειες *(περιοχές)* ή πολύγωνα σε διάφορες μορφές.

Ένα σημείο περιγράφει ένα αντικείμενο του οποίου μας ενδιαφέρει η θέση, π.χ. μια πόλη σε χάρτη μεγάλης κλίμακας. Μια γραμμή*(καμπύλη)* που περιγράφει τη μετακίνηση στο χώρο ή τις διασυνδέσεις στο χώρο*(δρόμοι, ποτάμια, κλπ.*). Μια περιοχή αντίστοιχα, είναι η παρουσίαση ενός αντικειμένου που έχει σχετική έκταση*(π.χ. δάσος ή λίμνη)*, μεγαλύτερη σε σχέση με ένα σημείο. Αυτοί οι όροι αναφέρονται συνήθως σε δισδιάστατο χώρο, αλλά ισχύουν και σε χώρους τριών ή περισσότερων διαστάσεων.

Σχήμα 1: Γεωμετρικοί τύποι δεδομένων
An Introduction to Spatial Database Systems - Ralf Hartmut Güting

Οι ακατέργαστες κινούμενες διαδρομές *(raw movement tracks)* μπορούν να χρησιμοποιηθούν όπως είναι, για περαιτέρω ανάλυση ή να μετατραπούν σε άλλα είδη που αναπαριστούν κίνηση, όπως οι σημασιολογικές *(semantic)* τροχιές.

1.3.1 Μοντελοποιήση Δεδομένων Κίνησης

Η κίνηση ενός αντικειμένου, μπορεί να απεικονιστεί ως μια συνάρτηση από το χρονικό πεδίο $I \subseteq R$ στο γεωγραφικό πεδίο $E \subseteq R^3$.

$$I \subseteq R \rightarrow S \subseteq R^3 : t \rightarrow l(t) = \big(l_x(t), l_y(t), l_z(t)\big)$$

Όπου $l(t)$, η τοποθεσία του αντικειμένου κατά τη χρονική στιγμή t. Έτσι η πραγματική τροχιά του κινούμενου αντικειμένου χαρακτηρίζεται ως

$$T_{act} = \{t, l(t) | t \in I\} \subset R^3 \times R$$

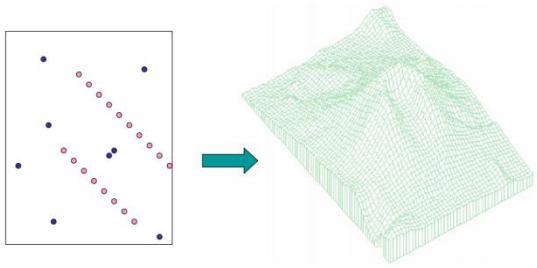
Η πραγματική τροχιά όμως, δεν μπορεί να αναπαρασταθεί πλήρως από τα υπολογιστικά συστήματα, αλλά αντί αυτού αναπαριστάται μία ακολουθία χώρο-χρονικών θέσεων του αντικειμένου.

$$T = \{(p_1, t_1), (p_2, t_2), \dots, (p_n, t_n)\}$$

Όπου $p_i \in R^3$, $t_i \in R, 1 \leq i \leq n$ *και* $t_1 \leq t_2 \leq \cdots \leq t_n$. Η αντιπαραβολή*(interpolation)* της τροχιάς T στην πραγματική τροχιά T_{act}, δεν είναι μοναδική, καθώς η καμπύλη της τροχιάς δημιουργείται κατά προσέγγιση με βάση τις ακολουθίες των χώρο-χρονικών θέσεων του αντικειμένου.

Η αντιπαραβολή χωρίζεται σε τέσσερις βασικές μορφές, βάση των γεωμετρικών τύπων δεδομένων που αναφέραμε προηγουμένως.

- Αντιπαραβολή σημείου σε σημείο*(τυχαία σημεία σε επιφάνεια πλέγματος)*.
- Αντιπαραβολή σημείου σε γραμμή*(τυχαία σημεία σε περιγράμματα γραμμών)*.
- Αντιπαραβολή γραμμής σε σημείο*(περιγράμματα σε επιφάνειες πλέγματος)*.
- Αντιπαραβολή περιοχής σε περιοχή.



Σχήμα 2: Αντιπαραβολή τυχαίων σημείων σε πλέγμα.
by Bahram Saghafian, Interpolation

Στη βιβλιογραφία, έχουν προταθεί πολλές μέθοδοι αντιπαραβολής, με πιο διαδεδομένη τη μέθοδο της Γραμμικής Αντιπαραβολής*(linear interpolation)*, η οποία εμφανίζεται επαρκής στις περισσότερες εφαρμογές και χρησιμοποιείται πιο συχνά λόγω της απλότητας της.

Σύμφωνα με τη μέθοδο της γραμμικής παρεμβολής, ακολουθίες χώρο-χρονικών θέσεων (p_i, t_i), συνδέονται σε ευθείες γραμμές στον τρισδιάστατο χώρο και η τοποθεσία του αντικειμένου στο (μη καταγεγραμμένο) χρόνο $t \in (t_i, t_{i+1})$, όπου t_i και t_{i+1}, διαδοχικές χρονικές στιγμές, υπολογίζεται ως εξής:

$$p(t) = \left(x_i + \frac{t - t_i}{t_{i+1} - t_i}(x_{i+1} - x_i), y_i + \frac{t - t_i}{t_{i+1} - t_i}(y_{i+1} - y_i) \right)$$

Στην παραπάνω εξίσωση, γίνεται η παραδοχή ότι η ταχύτητα και η κατεύθυνση του αντικειμένου, παραμένουν σταθερές, στο χρονικό διάστημα $[t_i, t_{i+1})$.

Άλλες πιο διαδεδομένες μέθοδοι αντιπαραβολής είναι:

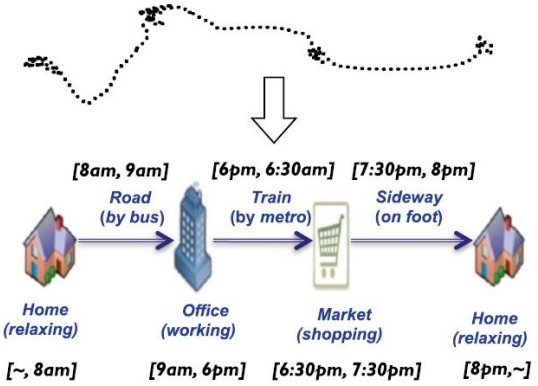
- Πολύγωνα του Thiessen
- Τριγωνισμός*(Triangulation)*
- Κινούμενος Μέσος
- B-Splines
- Trend Surfaces
- Kriging

Ο αναγνώστης μπορεί να ανατρέξει για περαιτέρω πληροφορία σχετικά, στο σύγγραμμα των (Pelekis & Theodoridis, 2014) καθώς και στο κείμενο του (Saghafian, 2013).

1.4 Σημασιολογικές Τροχιές σε Δεδομένα Κίνησης

Με τον όρο σημασιολογική τροχιά, αναφερόμαστε σε μία τροχιά, η οποία είναι εμπλουτισμένη με πρόσθετη χρήσιμη πληροφορία, πέρα από την κίνηση αυτή κάθε αυτή του αντικειμένου. (Pelekis & Theodoridis, 2014) Έτσι, μπορούμε να θεωρήσουμε ως σημασιολογική τροχιά, ως μία ακολουθία από υπό-τροχιές ή επεισόδια στάσεων *(Stops)* και κινήσεων *(Move)*, αποτελούμενα από ετικέτες *(tags)*, όπου:

- Στάση, όταν το αντικείμενο μελέτης παραμένει στατικό σε μία περιοχή.
- Κίνηση, η κίνηση που κάνει το αντικείμενο μελέτης, μεταξύ δύο στάσεων.
- Ετικέτες, αποτελούν μετά-δεδομένα, τα οποία σχετίζονται με στάσεις και κινήσεις.



Σχήμα 3: Ακατέργαστη vs Σημασιολογική Τροχιάς
Mobility Data Management and Exploration by N. Pelekis, Y. Theodoridis

Η προσθήκη στοιχείων στις ακατέργαστες τροχιές είναι μια διαδικασία που ονομάζεται σημασιολογικός εμπλουτισμός *(semantic enrichment)*. Ο εμπλουτισμός *(enrichment)* εκφράζει την ιδέα ότι τα υπάρχοντα στοιχεία συμπληρώνονται με πρόσθετα στοιχεία, που ονομάζονται σχολιασμοί *(annotations)*, όπου ένας σχολιασμός, μπορεί να είναι συνδεδεμένος σε μία τροχιά στο σύνολο της, κάτι το οποίο όμως συνήθως δεν είναι ιδιαίτερα χρήσιμο, αφού μπορεί να μας οδηγήσει σε μία σειρά από επαναλαμβανόμενους σχολιασμούς στην τροχιά. Έτσι το πιο σύνηθες, είναι να χωρίζουμε την τροχιά σε υπό-τροχιές ή επεισόδια, σε ορισμένα σημεία με ιδιαίτερη σημασία και να προσθέτουμε σχολιασμούς στα επεισόδια αυτά.

Ο σχολιασμός των επεισοδίων αυτών, απαιτεί μία προ-επεξεργασία, όσον αφορά τον εντοπισμό τους στην τροχιά. Η διαδικασία αυτή, αναφέρεται στην βιβλιογραφία ως τμηματοποίηση τροχιάς (*trajectory segmentation)*, όπου κάθε τμήμα πληροί κάποια κριτήρια. Η τμηματοποίηση λέμε ότι είναι η βέλτιστη, όταν επιτυγχάνουμε τον ελάχιστο αριθμό τμημάτων στην τροχιά, ικανοποιώντας πλήρως τα κριτήρια που έχουμε θέσει. Το πιο συνηθισμένο κριτήριο, αφορά το αν το αντικείμενο μελέτης μας είναι σταθμευμένο ή κινείται και πάνω σε αυτό το κριτήριο, έχουν δημιουργηθεί διάφοροι αλγόριθμοι για την ανακάλυψη στάσεων σε μία τροχιά.

Η τμηματοποίηση, διακρίνεται σε διακριτή και συνεχής. Ως διακριτή τμηματοποίηση, νοούμε τον διαχωρισμό μιας διακριτής τροχιάς, όπου ένα τμήμα αποτελείται από συνεχόμενα χρονικά ορόσημα *(timestamps)*. Ενώ ως συνεχής τμηματοποίηση, είναι ο διαχωρισμός μιας συνεχής τροχιάς σε τμήματα, όπου κάθε τμήμα, αποτελείται από υπό-τροχιές οι οποίες μπορούν να ξεκινούν και να τελειώνουν οπουδήποτε.

Η τμηματοποίηση, αποτελεί μία μέθοδος που χρησιμοποιείται συχνά στην ανάλυση χρονοσειρών. Ο στόχος της είναι η κατανομή των δεδομένων σε ομοιογενή τμήματα, όπου τα δεδομένα

αντιπροσωπεύονται ικανοποιητικά, όπως και η μείωση του θορύβου των δεδομένων με τελικό σκοπό την κατανόηση και πρόβλεψη της συμπεριφοράς των δεδομένων της χρονοσειράς.

Οι μέθοδοι της τμηματοποίησης και του εμπλουτισμού μιας τροχιάς, αποτελούν ένα πολύ σημαντικό κομμάτι στην μελέτη των χωρικών δεδομένων, αφού υποδηλώνουν την ύπαρξη σημείων ενδιαφέροντος *(places of interest – POIs)* πάνω στο χώρο. Τα σημεία αυτά, είναι πολύ χρήσιμα για την εξαγωγή συμπερασμάτων αναφορικά με τη συμπεριφορά ενός κινούμενου αντικειμένου, όμως δεν θα αναφερθούμε περαιτέρω, καθώς δεν είναι αυτός ο σκοπός της εργασία. Παρόλα αυτά, υπάρχουν αρκετές εργασίες που αναφέρονται στο κομμάτι αυτό, στις οποίες μπορεί να ανατρέξει ο αναγνώστης. (Buchin, Driemel, Kreveld, & Sacristan, 2011) (Spinsanti, Celli, & Renso, 2010)

ΚΕΦΑΛΑΙΟ 2

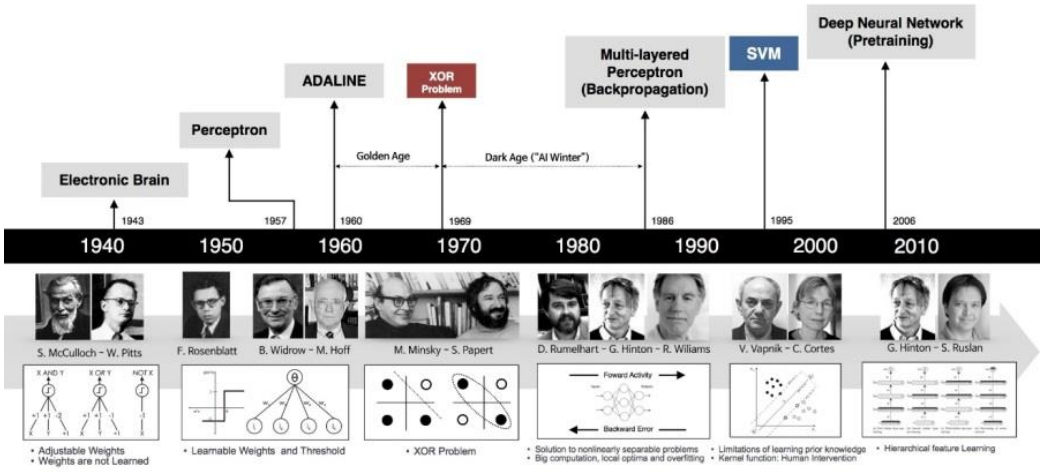
Εισαγωγή στη Βαθιά Μάθηση *(Deep Learning)*

Στο κεφάλαιο αυτό, σκοπός μας είναι να κάνουμε μία εισαγωγή στην Βαθιά Μάθηση. Ξεκινάμε την εισαγωγή αυτή με την παρουσίαση μερικών ιστορικών στοιχείων και στη συνέχεια παρουσιάζουμε τις διαφορές της με διάφορους άλλους τομείς της επιστήμης, αλλά και την χρησιμότητα που έχει ή ενδέχεται να έχει η Βαθιά Μάθηση στο εγγύς μέλλον.

2.1 Μερικά Ιστορικά Στοιχεία

Μέχρι πρόσφατα, οι περισσότερες τεχνικές μηχανικής μάθησης και επεξεργασίας σήματος εκμεταλλεύονταν ρηχές *(shallow-structured)* αρχιτεκτονικές. Αυτές οι αρχιτεκτονικές τυπικά περιέχουν το πολύ ένα ή δύο στρώματα μετασχηματισμών μη γραμμικών χαρακτηριστικών. Χαρακτηριστικά παραδείγματα αποτελούν τα μοντέλα Λογιστικής Παλινδρόμησης, Μέγιστης Εντροπίας *(MaxEnt)*, Support Vector Machine *(SVM)* και πολυεπίπεδα Perceptrons *(MLP)* με ένα κρυμμένο στρώμα. Αυτές οι ρηχές αρχιτεκτονικές αποδείχθηκαν αρκετά αποτελεσματικές σε πολλά απλά και καλά δομημένα προβλήματα, αλλά λόγω της περιορισμένης ισχύς μοντελοποίησης τους, παρουσιάζουν αρκετά μεγάλη δυσκολία σε πιο σύνθετα καθημερινά προβλήματα.

Η ανάπτυξη των τεχνητών νευρωνικών δικτύων συνδέεται στενά με την ανάπτυξη στις επιστήμες της Βιολογίας και της Νευρολογίας. Έτσι, οι πρώτες ενδείξεις για την ανάγκη επεξεργασίας της πληροφορίας σε πολλαπλά επίπεδα, προήλθε από τις επιστήμες αυτές, όπου έχει παρατηρηθεί ότι το ανθρώπινο νευρικό σύστημα, αποτελείται από πολλαπλά επίπεδα επεξεργασίας.



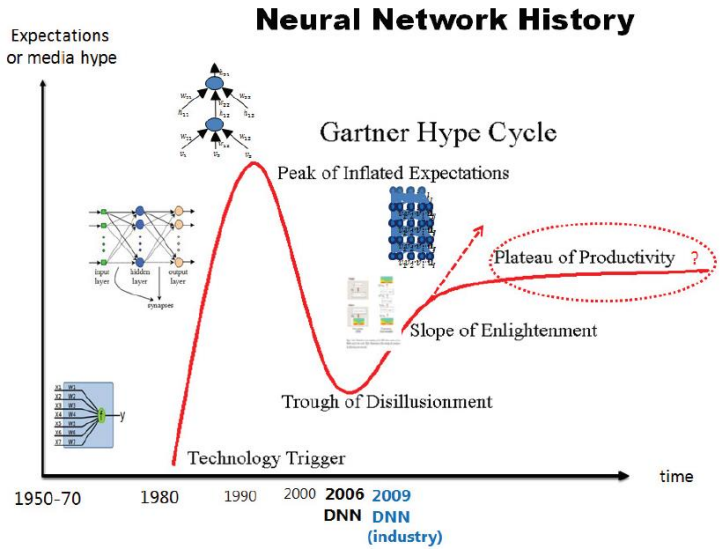
Σχήμα 4: Χρονολογίες ορόσημα στην ανάπτυξη νευρωνικών δικτύων
By Andrew L. Beam - machine learning and medicine.

Η ιδέα της δημιουργίας μιας «μηχανής σκέψης» ξεκινάει το 1950 με τον Alan Turing, ο οποίος στο βασικό του έγγραφο «Υπολογιστικές Μηχανές και Νοημοσύνη» έθεσε διάφορα κριτήρια για να κρίνει αν μια μηχανή θα μπορούσε να ειπωθεί ως έξυπνη, η οποία από τότε έγινε γνωστή ως «δοκιμή Turing». Η ιστορία της Βαθιάς Μάθησης όμως, ουσιαστικά ξεκινάει από το 1943, όταν ο Walter Pitts και ο Warren McCulloch δημιούργησαν ένα μοντέλο υπολογιστή βασισμένο στα νευρωνικά δίκτυα του ανθρώπινου εγκεφάλου. Χρησιμοποίησαν έναν συνδυασμό αλγορίθμων και μαθηματικών που ονόμαζαν «λογική κατωφλίου» για να μιμηθούν τη διαδικασία σκέψης. Δεν είναι όμως, μέχρι το "perceptron" του Frank Rosenblatt το 1959 όπου βλέπουμε τον πρώτο πραγματικό πρόδρομο των σύγχρονων νευρωνικών δικτύων. Από τότε, η Βαθιά Μάθηση έχει εξελιχθεί σταθερά, με μόνο δύο σημαντικά διαλείμματα στην ανάπτυξή της. Και οι δύο ήταν συνδεδεμένοι με τους περίφημους «χειμώνες» της τεχνητής νοημοσύνης. (Beam, 2017)

Ο πρώτος «χειμώνας», ήρθε όταν ο Marvin Minsky, ο οποίος συχνά θεωρείται ένας από τους πατέρες της τεχνητής νοημοσύνης, απέδειξε μαζί με τον Seymor Papert ότι το "perceptron" του Rosenblatt δεν ήταν ικανό να μάθει την απλή συνάρτηση XOR, όσο και να το εκπαιδεύαν. Αυτό, τη σημερινή εποχή δεν αποτελεί έκπληξη, καθώς το μοντέλο του perceptron είναι γραμμικό και η συνάρτηση XOR είναι μη γραμμική, αλλά εκείνη την εποχή ήταν αρκετή αυτή η διαπίστωση ώστε να «σκοτώσει» όλη την έρευνα για τα νευρωνικά δίκτυα έχοντας ως αποτέλεσμα τον πρώτο «χειμώνα» της τεχνητής νοημοσύνης.

Στην πορεία το 1980, προτάθηκε από τον Kunihiko Fukushima, ένα ιεραρχικό τεχνητό νευρωνικό δίκτυο, το Neocognitron, το οποίο θεωρείται ο πρόγονος των σημερινών βαθιών νευρωνικών δικτύων. Αν και η δομή του έμοιαζε με των σημερινών, ο τρόπος μάθησης ήταν διαφορετικός και σε συνδυασμό με την μη επαρκή υπολογιστική δύναμη της εποχής, οδήγησε σε μη επιτυχή αποτελέσματα.

Το 1986, ο Geoff Hinton μαζί με τους David Rumelhart και Ronald Williams, παρουσίασαν μία εργασία, όπου απέδειξαν ότι τα νευρωνικά δίκτυα με πολλά κρυμμένα στρώματα, θα μπορούν να εκπαιδεύονται αποτελεσματικά με μία σχετικά απλή διαδικασία. Ήταν η μέθοδος Back-propagation, όπου θα επέτρεπε στα νευρωνικά δίκτυα να ξεπεράσουν την αδυναμία του perceptron, αφού τα πρόσθετα στρώματα προμήθευαν το δίκτυο με τη δυνατότητα να μάθουν μη γραμμικές λειτουργίες. Στην πορεία αποδείχθηκε ότι είχαν την ικανότητα να μαθαίνουν οποιαδήποτε συνάρτηση. Αυτό το αποτέλεσμα είναι γνωστό ως θεώρημα γενικής προσέγγισης *(universal approximation theorem)*.



Σχήμα 5: Η πορεία της εξέλιξης των τεχνητών νευρωνικών δικτύων σε σχέση και με τις προσδοκίες στον χρόνο.
By Li Deng and Dong Yu - Deep Learning Methods and Applications

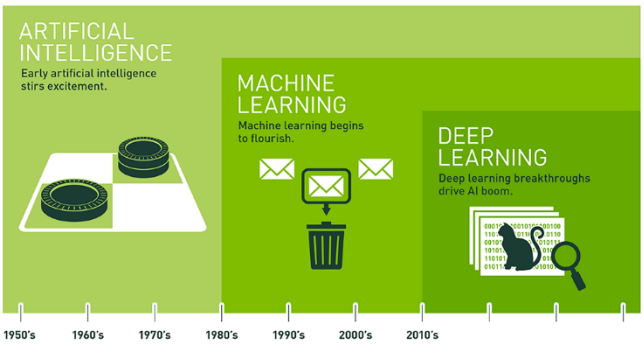
Τα επόμενα χρόνια υπήρξε μία περίοδος, όπου λόγω και της υπερβολικής φιλοδοξίας των τότε επιστημόνων, είχε σαν αποτέλεσμα να υπερβάλλουν σε σχέση με το άμεσο δυναμικό *(potential)* των νευρονικών δικτύων αλλά και της Τεχνητής Νοημοσύνης γενικότερα και σηματοδότησε τον δεύτερο «χειμώνα» της Τεχνητής Νοημοσύνης. Αποτέλεσμα αυτού ήταν ότι η επιστήμη της τεχνητής νοημοσύνης να φθάσει στα όρια της ψευδοεπιστήμης. Ευτυχώς όμως ορισμένοι επιστήμονες συνέχισαν να εργάζονται στον τομέα αυτόν για να ανατρέψουν την κατάσταση και το 1995 η Corinna Cortes και ο Vladimir Vapnik, ανέπτυξαν τα Support Vector Machines *(SVM)*.

Το επόμενο σημαντικό βήμα στην εξέλιξη της Βαθιάς Γνώσης έγινε το 1999 με την ανάπτυξη πλέον ταχύτερων επεξεργαστών και καρτών γραφικών για τους υπολογιστές. Στην περίοδο αυτή τα τεχνητά νευρωνικά δίκτυα ξεκίνησαν να συναγωνίζονται τα SVMs, αφού παρήγαγαν πολύ καλύτερα αποτελέσματα αν και ήταν πολύ πιο αργά σε σχέση με αυτά. (Foote, 2017)

Επί του παρόντος η επεξεργασία των Μεγάλων Δεδομένων *(Big Data)* και της Τεχνητής Νοημοσύνης, εξαρτώνται από την Βαθιά Μάθηση, η οποία συνεχίζει να εξελίσσεται με ταχύτατους ρυθμούς.

2.2 Η διαφορά Τεχνητής Νοημοσύνης, Μηχανικής Μάθησης και Βαθιάς Μάθησης

Μπορούμε να σκεφτούμε τη βαθιά μάθηση, τη μηχανική μάθηση και την τεχνητή νοημοσύνη ως ένα σύνολο το ένα μέσα στο άλλο, ξεκινώντας από το μικρότερο και το πιο εξειδικευμένο. Με την έννοια ότι η βαθιά μάθηση είναι ένα υποσύνολο της μηχανικής μάθησης και η μηχανική μάθηση είναι ένα υποσύνολο του τεχνητής νοημοσύνης, το οποίο αποτελεί όρο ομπρέλα για οποιοδήποτε πρόγραμμα υπολογιστή που κάνει κάτι έξυπνο. Με άλλα λόγια, όλα τα προβλήματα μηχανικής μάθησης είναι τεχνητή νοημοσύνη, αλλά όλα τα προβλήματα τεχνητής νοημοσύνης δεν είναι προβλήματα μηχανικής μάθησης, και ούτω καθεξής. (Copeland, 2016)



Σχήμα 6: Γράφημα της εξέλιξης και της εξειδίκευσης του AI στο χρόνο
By Michael Copeland

Ως Τεχνητή Νοημοσύνη, μπορούμε να θεωρήσουμε την προσπάθεια για την προσομοίωση ανθρώπινης εφυούς συμπεριφοράς από τους υπολογιστές. Δηλαδή, η προσπάθεια μέσω υπολογιστών να εκτελέσουμε καθήκοντα που συνήθως απαιτούν ανθρώπινη νοημοσύνη, όπως η οπτική αντίληψη, η αναγνώριση ομιλίας, η λήψη αποφάσεων και η μετάφραση μεταξύ των γλωσσών. Η μηχανική μάθηση, όπως αναφέραμε προηγουμένως αποτελεί ένα υποσύνολο της τεχνητής νοημοσύνης. Μια πτυχή που τις διαχωρίζει είναι ότι η μηχανική μάθηση είναι δυναμική, δηλαδή έχει τη δυνατότητα να τροποποιείται αυτόματα χωρίς την ανθρώπινη παρέμβαση, συνήθως όταν «εκτείθεται» σε περισσότερα δεδομένα. Αντίστοιχα, η Βαθιά Μάθηση αποτελεί υποσύνολο της μηχανικής μάθησης και συνήθως όταν αναφερόμαστε σε βαθιά μάθηση, αναφερόμαστε στα βαθιά τεχνητά νευρωνικά δίκτυα, τα οποία είναι

και το επίκεντρο της παρούσας εργασίας. Η διαφορά τους, έγκειται στο γεγονός ότι η βαθιά μάθηση, χρησιμοποιεί πολλαπλά στρώμματα επεξεργασίας των δεδομένων για καλύτερη και πιο αποδοτική διακπεραίωση των προβλημάτων.

2.3 Η σχέση με τη Στατιστική

Η μηχανική μάθηση και η στατιστική, έχουν συνυπάρξει για μεγάλο χρονικό διάστημα και συχνά τίθεται το ερώτημα για το ποιες είναι οι ομοιότητες τους και ποιες οι διαφορές τους. Η μηχανική μάθηση είναι σχετικά νέο επιστημονικό πεδίο, ενώ η στατιστική επικρατεί για μεγάλο χρονικό διάστημα, έτσι είναι εύλογο, η μηχανική μάθηση να έχει υιοθετήσει πολλές μεθόδους από την στατιστική, αφού ο σκοπός και των δύο, είναι σχεδόν ο ίδιος.

Τόσο η στατιστική όσο και η μηχανική μάθηση δημιουργούν μοντέλα από δεδομένα, αλλά για διαφορετικούς σκοπούς. Η μηχανική μάθηση αφορά περισσότερο την πρόβλεψη σε αντίθεση με την στατιστική που στοχεύει περισσότερο στην ανάλυση.

Άλλο ένα βασικό χαρακτηριστικό, το οποίο διαχωρίζει τα δύο πεδία, είναι ο όγκος των δεδομένων. Η μηχανική μάθηση εξελίχθηκε ραγδαία τα τελευταία χρόνια λόγω και της ισχύς των νέων υπολογιστών και πλέον μπορεί και επεξεργάζεται δεδομένα με εκατοντάδες παραμέτρους σε πολύ μικρό χρονικό διάστημα. Είναι δυνατόν πλέον και με στατιστικές μεθοδους να επεξεργαστούμε μεγάλο αριθμό δεδομένων, αλλα αυτό δεν αποτελεί αναγκαιότητα, σε αντίθεση με τους αλγορίθμους μηχανικής μάθησης, όπου η διαθεσιμότητα μεγάλου όγκου ιστορικών δεδομένων, είναι απαραίτητη.

Υπάρχει επίσης μια ουσιαστική διαφορά στον αριθμό των υποθέσεων στις οποίες λειτουργούν τα δύο αυτά εργαλεία. Η στατιστική μοντελοποίηση, για παράδειγμα, δουλεύει με πολλές υποθέσεις σε σύγκριση με τους αλγόριθμους μηχανικής μάθησης. Είτε πρόκειται για γραμμική παλινδρόμηση είτε για λογιστική παλινδρομήση, απαιτούν το δικό τους σύνολο υποθέσεων. Για παράδειγμα, μια γραμμική παλινδρόμηση υποθέτει ότι υπάρχει ελάχιστη ή καθόλου πολυσυγραμμικότητα στα δεδομένα, γραμμική σχέση μεταξύ ανεξάρτητης και εξαρτημένης μεταβλητής και ομοσεσκεδαστικότητα μεταξύ άλλων. Η μηχανική μάθηση από την άλλη πλευρά δεν βασίζεται σε μεγάλο βαθμό σε αυτές τις υποθέσεις. Ο αλγόριθμος δεν απαιτεί την κατανομή της εξαρτημένης ή ανεξάρτητης μεταβλητής που θα καθοριστεί.

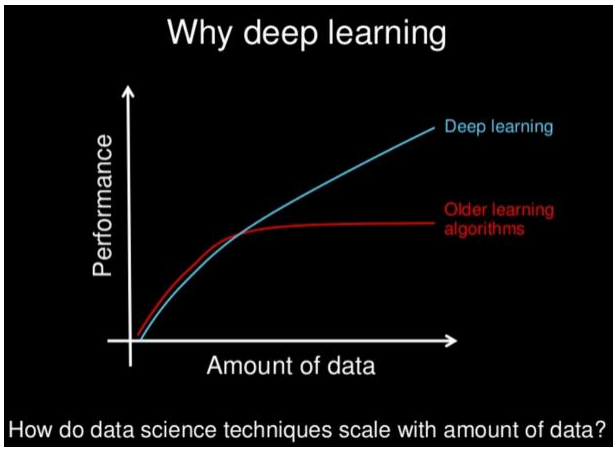
Τέλος, άλλη μια διαφορά που μπορούμε να συμπεράνουμε, είναι αυτή του ανθρώπινου φόρτου που απαιτείται από τις δύο αυτές τεχνικές. Η μηχανική μάθηση λειτουργεί σε επαναλήψεις και προσπαθεί να βρει μοτίβα κρυμμένα σε δεδομένα, ξανά και ξανά. Αυτό απαιτεί όσο το δυνατόν λιγότερη ανθρώπινη εξάρτηση για την επίτευξη καλύτερων αποτελεσμάτων. Η εκμάθηση μηχανών αξιολογεί πολλά δεδομένα και είναι ανεξάρτητη από τις υποθέσεις, η δύναμη πρόβλεψης είναι ισχυρή για τα μοντέλα αυτά, μειώνοντας δραστικά τις ανθρώπινες προσπάθειες. Όπως γνωρίζουμε, όσο λιγότερες είναι οι υποθέσεις, τόσο υψηλότερη είναι η προγνωστική δύναμη. Η στατιστική από την άλλη πλευρά, περιλαμβάνει μοντέλα που βασίζονται έντονα στα μαθηματικά και στην εκτίμηση των συντελεστών. Επιπλεόν, απαιτεί από τον στατιστικό να καταλάβει τη σχέση μεταξύ της μεταβλητής πριν την τοποθέτησή της και κατά συνέπεια περισσότερη ανθρώπινη προσπάθεια. (Deoras, 2017)

2.4 Γιατί Βαθιά Μάθηση?

Υπάρχουν πολλοί λόγοι για να ασχοληθούμε με την βαθιά μάθηση και συγκεκριμένα με τα νευρωνικά δίκτυα. Ένας βασικός λόγος είναι για να αρχίσουμε να κατανοούμε το πώς ο ανθρώπινος εγκέφαλος λειτουργεί και συγκεκριμένα οι εφαρμογές εμπνευσμένες από αυτόν και να καταφέρουμε να λύσουμε

πρακτικά προβλήματα της καθημερινότητας. Η βασική πηγή της δύναμης της βαθιάς μάθησης, έγκειται στο γεγονός ότι οι αλγόριθμοι μπορούν συνεχώς να εκπαιδεύονται με νέα δεδομένα και ως εκ τουτου να συνεχίσουν να βελτιώνονται και να παράγουν καλύτερα αποτελέσματα, μπορούμε δηλαδή να πούμε ότι αποκτούν «εμπειρία», όπως ο ανθρώπινος οργανισμός.

Ο λόγος που η βαθιά μάθηση αρχίζει και κερδίζει όλο και περισσότερο έδαφος, αφορά δύο τομείς, την υπολογιστική ισχύ και τον μεγάλο όγκο δεδομένων. Βρισκόμαστε στην εποχή των μεγάλων δεδομένων, όπου χρησιμοποιούμε ολο και αυξανόμενα σετ δεδομένων με αποτέλεσμα πολλές τεχνικές να μην μπορούν να αποδώσουν κατάλληλα ή να απαιτείται μεγάλο φόρτος εργασίας. Επίσης οι αλγόριθμοι βαθιάς μάθησης, έχουν αρχίσει και χρησιμοποιούνται περισσότερο λόγω και της δυνατοτητας που παρέχεται πλέον από τους σύχρονους υπολογιστές, μία δυνατότητα που δεν υπήρχε τα προηγουμενα χρόνια. Έτσι υπερτερούν από πολλές τεχνικές μηχανικής μάθησης όσον αφορά την αποτελεσματικότητα και την αποδοτικότητα τους σε όλο και περισσότερες εφαρμογές, λόγω και της χρήσης πολλαπλών στρωμμάτων στην επεξεργασία των δεδομένων. Τέλος, μπορούμε να πούμε ότι οι αλγόριθμοι βαθιάς μάθησης είναι πιο εύχρηστοι από πολλούς αλγόριθμους μηχανικής μάθησης. (Woodie, 2017)



Σχήμα 7 By Andrew Ng

2.5 Εφαρμογές Βαθιάς Μάθησης.

Η βαθιά μάθηση, έχει αναπτυχθεί ραγδαία τα τελευταία έτη και η χρήση της, έχει δώσει αρκετές λύσεις στην αντιμετώπιση πολλών σύγχρονων προβλημάτων. (Ρεφανίδης, 2011) Ορισμένοι κλάδοι στους οποίους έδωσε λύση ή αναμένεται να δώσει στη πορεία είναι τα

- Χρηματοοικονομικά
  - Πρόβλεψη συναλλαγματικών μεταβολών
  - Διαχείριση χαρτοφυλακίων
  - Αξιολόγηση αιτήσεων για δάνειο
  - Αποτίμηση ακίνητης περιουσίας

- Υγεία
  - Ανάλυση συμπτωμάτων και διάγνωση για διάφορες ασθένειες

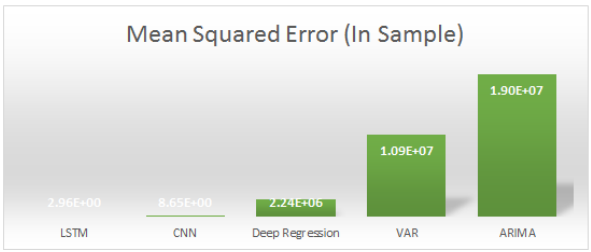
- Τηλεπικοινωνίες
  - Marketing
  - Συμπίεση δεδομένων
  - Μετάφραση γλώσσας σε πραγματικό χρόνο

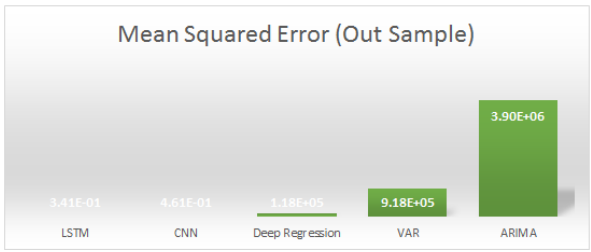
- Τεχνολογία
  - Ρομποτική
  - Μεταφορές
  - Κατασκευές
  - Αεροδιαστημική

Στα χρηματοοικονομικά η χρήση της βαθιάς μάθησης έχει αποδειχθεί ότι έχει πολύ καλά αποτελέσματα τόσο στην πρόβλεψη των τιμών στο χρηματιστήριο με τα Recurrent Neural Networks όσο και στη κατασκευή χαρτοφυλακίων. Μάλιστα ύστερα από έρευνα, έχει δειχθεί ότι έχει καλύτερα αποτελέσματα από παραδοσιακές μεθόδους στην πρόβλεψη τιμών, όπως είναι η ARIMA, όπως φαίνεται και στην εικόνα παρακάτω.

Model	Mean Squared Error (In Sample)
LSTM	2.96E+00
CNN	8.65E+00
Deep Regression	2.24E+06
VAR	1.09E+07
ARIMA	1.90E+07

Model	Mean Squared Error (Out Sample)
LSTM	3.41E-01
CNN	4.61E-01
Deep Regression	1.18E+05
VAR	9.18E+05
ARIMA	3.90E+06





Σχήμα 8: Σύγκριση Διαφορετικών Μεθόδων By Sonam Srivastava

Όσον αφορά τον τομέα της υγείας, η χρήση των νευρωνικών δικτύων, έχει αρχίσει ήδη να δημιουργεί εκπληκτικά αποτελέσματα, λόγω και της ικανότητας τους να ανιχνεύουν πρότυπα. Για αυτό, έχουν γίνει μελέτες όπου φαίνεται ότι αποδίδουν πολύ καλά στη πρόωρη διάγνωση σημαντικών ασθενειών που πλήττουν την ανθρωπότητα, όπως ο καρκίνος. Φαίνεται λοιπόν, ότι αλλάζουν σιγά σιγά τον τρόπο με τον οποίο οι γιατροί καταφέρνουν να διαγνώσουν τις ασθένειες, καθιστώντας τα διαγνωστικά ταχύτερα, φθηνότερα και πιο ακριβή από ποτέ.

Στον τομέα των τηλεπικοινωνιών, έχουν ήδη αναδειχθεί οι δυνατότητες τους, αφού η χρήση τους είναι φανερή σε πολλές πτυχές τις καθημερινότητάς μας. Οι διαφημίσεις που κατακλύζουν το διαδίκτυο, όταν προσπαθούμε να αναζητήσουμε ένα προϊόν μέσω αυτού, είναι ένα μικρό μόνο δείγμα των δυνατοτήτων τους.

Όσον αφορά την τεχνολογική πρόοδο, τα νευρωνικά δίκτυα αν και μέρος αυτής, δίνουν λύσεις και σε επιμέρους τομείς. Μερικοί από αυτούς είναι η ρομποτική, η ανάγνωση και μετάφραση κειμένων, καθώς

και η συνεισφορά τους σε τεχνολογίες που αφορούν την ανίχνευση και πρόβλεψη τροχιών που θα ασχοληθούμε εκτενέστερα και στα πλάισια αυτής της εργασίας.

Υπάρχουν σαφώς και πολλοί άλλοι τομείς που έχουν βοηθήσει και καταλαβαίνουμε ότι η δυνατότητα που έχουν στην ανίχνευση προτύπων και συμπεριφορών στα δεδομένα που αναλύουν μέσα από την κατάλληλη εκπαίδευσή τους, τα καθιστούν ικανά να δώσουν πολλές λύσεις σε διάφορους τομείς, οι οποίοι όλο και θα αυξάνονται με την πάροδο του χρόνου.

ΚΕΦΑΛΑΙΟ 3

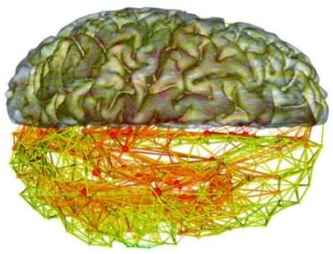
Μηχανική Μάθηση και Νευρωνικά Δίκτυα

Στο κεφάλαιο αυτό, θα μπούμε στην ουσία των νευρωνικών δικτύων. Ξεκινάμε με το πώς πρωτοήρθε η ιδέα δημιουργίας των τεχνητών νευρωνικών δικτύων και στη συνέχεια παρουσιάζουμε τον τρόπο με τον οποίο λειτουργούν και βελτιστοποιούνται. Για τη συγγραφή του κεφαλαίου αυτού, μεγάλη βαρύτητα δόθηκε στην εργασία του (Nielsen, 2015).

3.1 Η έννοια του τεχνητού νευρώνα και σύνδεση με τον βιολογικό

Η ιδέα ότι ο ανθρώπινος εγκέφαλος είναι ένας υπολογιστής, έχει φέρει στο προσκήνιο της γνωσιακής επιστήμης μια σειρά από θέματα που έχουν να κάνουν με την εξελικτική θεωρία και φυσικά την εξέλιξη του εγκεφάλου. Τα Τεχνητά Νευρωνικά Δίκτυα (ΤΝΔ), ξεκίνησαν ως μία προσπάθεια μοντελοποιήσης της συμπεριφοράς του ανθρώπινου εγκεφάλου, ωστόσο σήμερα η εξέλιξη τους είναι σχεδόν ανεξάρτητη, παρόλα αυτά, ομοιότητες μπορούν ακόμα να εντοπιστούν.

Οι βιολογικοί νευρώνες, όπως απεικονίζονται στο *Σχήμα 3.1.1,* αποτελούνται από τρία βασικά τμήματα που είναι το σώμα, ο άξονας και οι δενδρίτες. Αναλυτικότερα οι δενδρίτες, λαμβάνουν σήματα από γειτονικούς νευρώνες, τα σήματα αυτά είναι ηλεκτρικοί παλμοί που διαδίδονται μεταξύ του άξονα του νευρώνα πομπού και των δενδριτών του νευρώνα δέκτη με την βοήθεια χημικών διεργασιών. Το σημείο των χημικών διεργασιών, όπου ο άξονας ενός νευρώνα μεταδίδει το σήμα στους δενδρίτες του επόμενου λέγεται σύναψη. Αναφορικά αυτές οι διεργασίες μεταβάλλουν τα εισερχόμενα σήματα αλλάζοντας τη συχνότητα τους. Στην συνέχεια το σώμα αθροίζει τα εισερχόμενα σήματα και όταν αρκετά σήματα έχουν ληφθεί αποστέλλει το επεξεργασμένο σήμα στους γειτονικούς του νευρώνες μέσω του άξονα και η διαδικασία ξεκινά ξανά. Έτσι κάθε νευρώνας δέχεται πολλά σήματα ως είσοδο και μετά την επεξεργασία τους διαδίδει μόνο ένα σε όλους τους νευρώνες με τους οποίους συνδέεται. (Πλεύρου, 2012)

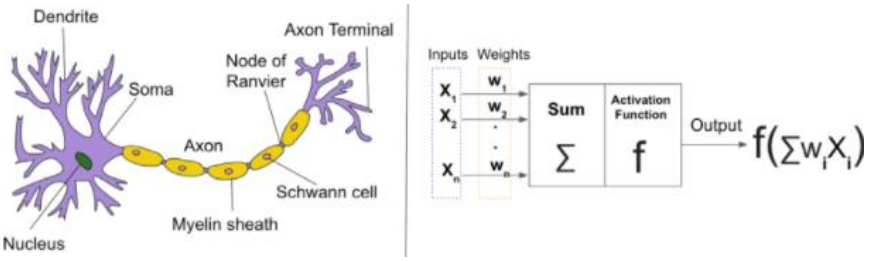


Σχήμα 9: Απεικόνιση Βιολογικών Νευρωνικών Δικτύων

Σε βιολογικά νευρωνικά δίκτυα όπως ο ανθρώπινος εγκέφαλος, η εκμάθηση επιτυγχάνεται με την πραγματοποίηση μικρών τροποποιήσεων σε μια υπάρχουσα αναπαράσταση, η διαμόρφωσή της οποίας περιέχει σημαντικές πληροφορίες. Τα πλεονεκτήματα των συνδέσεων μεταξύ των νευρώνων ή των βαρών

δεν ξεκινούν ως τυχαία, ούτε και η δομή των συνδέσεων, όπως συμβαίνει συνήθως στα τεχνητά νευρωνικά δίκτυα. Αυτή η αρχική κατάσταση είναι εν μέρει γενετική και είναι το υποπροϊόν της εξέλιξης. Με την πάροδο του χρόνου, το δίκτυο μαθαίνει πώς να εκτελεί νέες λειτουργίες προσαρμόζοντας τόσο την τοπολογία όσο και το βάρος των συνδέσεων των νευρώνων.

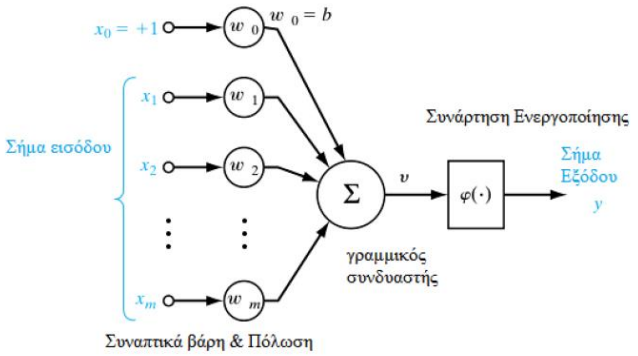
Σε αντίθεση με τα Βιολογικά Νευρωνικά Δίκτυα, τα Τεχνητά, συνήθως εκπαιδεύονται από το μηδέν, χρησιμοποιώντας μια σταθερή τοπολογία που επιλέγεται για το συγκεκριμένο πρόβλημα. Προς το παρόν, οι τοπολογίες τους δεν αλλάζουν με την πάροδο του χρόνου και τα βάρη τυχαία αρχικοποιούνται και ρυθμίζονται μέσω ενός αλγορίθμου βελτιστοποίησης για να χαρτογραφούν τις συνθέσεις των ερεθισμάτων εισόδου σε μια επιθυμητή συνάρτηση εξόδου. Ωστόσο, τα ΤΝΔ μπορούν επίσης να μάθουν με βάση μια προϋπάρχουσα κατάσταση. Αυτή η διαδικασία συνίσταται στην προσαρμογή των βαρών από μια προ-εκπαιδευμένη τοπολογία δικτύου σε ένα σχετικά αργό ρυθμό εκμάθησης για να αποδίδει καλά στα πρόσφατα παρεχόμενα δεδομένα κατάρτισης εισόδου.



Σχήμα 10: Αριστερά: Απεικόνιση Βιολογικού Νευρωνικού Δικτύου, Δεξιά: Απεικόνιση Τεχνητού Νευρωνικού Δικτύου. By Wikipedia

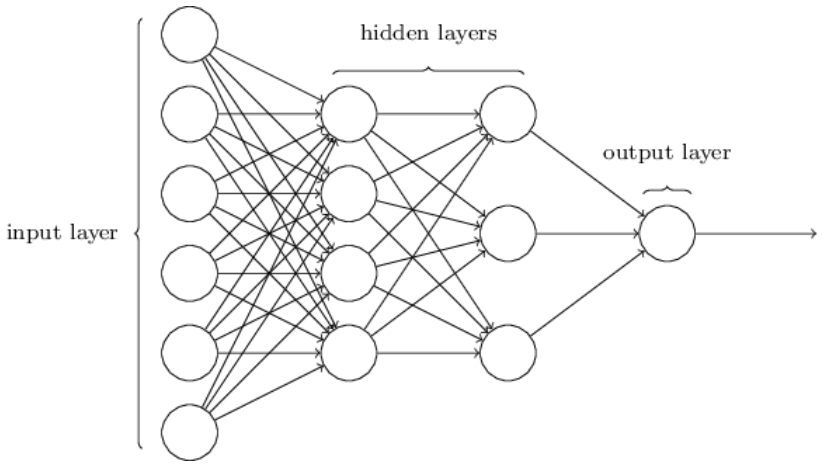
3.1.2 Το μοντέλο του Τεχνητού Νευρώνα

Το 1943 οι Warren McCulloch και Walter Pitts, δημιούργησαν ένα υπολογιστικό μοντέλο για νευρωνικά δίκτυα, βασισμένο σε μαθηματικά και αλγορίθμους και το ονόμασαν λογική κατωφλιού *(threshold logic)*. Η κατάσταση ενός τεχνητού νευρώνα περιγράφεται από έναν δυαδικό αριθμό y, όπου όταν $y = 0$, ο νευρώνας είναι αδρανής και αντίστοιχα όταν $y = 1$, ο νευρώνας ενεργοποιείται. Η βασική δομή του, φαίνεται στο *σχήμα 11* . (Nielsen, 2015)



Σχήμα 11: Το μοντέλο ενός τεχνητού νευρώνα. (Τριανταφυλιδου, 2016)

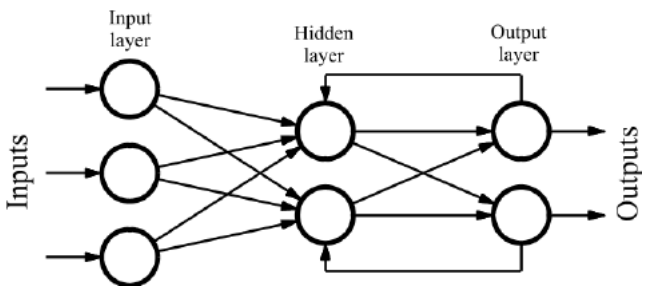
Συνδυάζοντας πολλούς νευρώνες μαζί κατασκευάζεται ένα νευρωνικό δίκτυο. Ένα νευρωνικό δίκτυο, αποτελείται από κόμβους με διασυνδεδεμένες συναπτικές συνδέσεις και συναρτήσεις ενεργοποίησης. Υπάρχουν τρεις τύποι νευρώνων: οι νευρώνες εισόδου, οι νευρώνες εξόδου και οι υπολογιστικοί νευρώνες ή κρυμμένοι νευρώνες, οι οποίοι βρίσκονται στο στρώμα εισόδου, το στρώμα εξόδου και το κρυμμένο στρώμα αντίστοιχα *(hidden layer)*. Ο όρος κρυμμένο, αναφέρεται στο ότι το στρώμα αυτό δεν είναι άμεσα ορατό από τα επίπεδα εισόδου και εξόδου. Τα σήματα εξόδου από ένα επίπεδο, χρησιμοποιούνται ως σήματα εισόδου για το επόμενο επίπεδο. Υπάρχουν νευρωνικά δίκτυα με περισσότερα από ένα κρυμμένο στρώμα, με τα οποία θα ασχοληθούμε και εκτενέστερα στην πορεία της παρούσας εργασίας. Ένα τέτοιο δίκτυο, παρουσιάζεται στο *σχήμα 12*, το οποίο ονομάζεται πλήρως συνδεδεμένο, με την έννοια ότι κάθε κόμβος ενός επιπέδου συνδέεται με κάθε άλλο κόμβο του επόμενου επιπέδου.



Σχήμα 12: Ένα πλήρες συνδεδεμένο νευρωνικό δίκτυο εμπρόσθιας τροφοδότησης

Τα ΤΝΔ κατηγοριοποιούνται ανάλογα με την αρχιτεκτονική τους και τον τρόπο με τον οποίο συνδέονται οι νευρώνες μεταξύ τους. Τα πιο συνηθισμένα δίκτυα ονομάζονται *feedforward* νευρωνικά δίκτυα, το οποίο σημαίνει ότι η πληροφορία εντός του δικτύου είναι πάντα προς τα εμπρός και δεν επιτρέπεται η προς τα πίσω τροφοδότηση του δικτύου.

Ωστόσο, υπάρχουν άλλα είδη νευρωνικών δικτύων, στα οποία επιτρέπεται η προς τα πίσω τροφοδότηση, τα οποία ονομάζονται *Recurrent Neural Networks (RNN)*. Με τα RNN θα ασχοληθούμε εκτενέστερα στην πορεία της εργασίας αυτής και κυρίως με ένα συγκεκριμένος είδος αυτών, τα Long Short – Term Memory *(LSTM)* δίκτυα. Σε αυτά τα δίκτυα οι αλγόριθμοι μάθησης είναι λιγότερο ισχυροί, παρόλα αυτά είναι πολύ πιο κοντά στα βιολογικά νευρωνικά δίκτυα από τα *feedforward* και μπορούν να δώσουν λύσεις σε ορισμένα προβλήματα στα οποία τα πρώτα δυσκολεύονται.



Σχήμα 13: Recurrent Neural Network
By Özel, T. and Davim J.P. (2009) Modeling and optimization of the machining processes and systems

Τέλος να αναφέρουμε ότι σε κάθε ΤΝΔ οι νευρώνες των γειτονικών επιμέρους επιπέδων, μπορούν είτε να συνδέονται όλοι με όλους, είτε ορισμένες συνδέσεις να μην υλοποιούνται. Έτσι τα ΤΝΔ μπορούν να

χαρακτηριστούν ως πλήρως συνδεδεμένα *(fully connected)* ή μερικώς συνδεδεμένα *(partially connected)* αντίστοιχα.

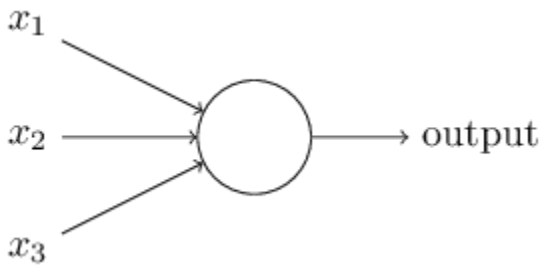
3.2 Συναρτήσεις Ενεργοποίησης

Οι συναρτήσεις ενεργοποίησης, χρησιμοποιούνται για να προσδιορίσουμε την τιμή εξόδου *(output)*, ενός νευρωνικού δικτύου. Αυτές χωρίζονται σε δύο ειδών συναρτήσεις, στις γραμμικές και στις μη-γραμμικές συναρτήσεις ενεργοποίησης. Στη συνέχεια θα παρουσιάσουμε μερικές από τις πιο βασικές μη-γραμμικές συναρτήσεις ενεργοποίησης, καθώς είναι και αυτές που χρησιμοποιούνται στην πράξη, αφού οι γραμμικές δεν βοηθάνε με την πολυπλοκότητα των συνηθισμένων δεδομένων που τροφοδοτούνται τα νευρωνικά δίκτυα.

3.2.1 Συνάρτηση Κατωφλιού - Perceptrons

Τα Perceptrons αναπτύχθηκαν τις δεκαετίες του 50’ και του 60’ από τον επιστήμονα Frank Rosenblatt. Σήμερα κυρίως χρησιμοποιούνται άλλα μοντέλα ΤΝΔ, το κυριότερο εκ των οποίων είναι ο Σιγμοειδές Νεύρωνας. Θα ξεκινήσουμε τη αναφορά μας όμως από τα perceptrons.

Ένα perceptron δέχεται πολλές δυαδικές εισροές $x_1, x_2, \dots, x_n$ και παράγει ένα μοναδικό δυαδικό output.



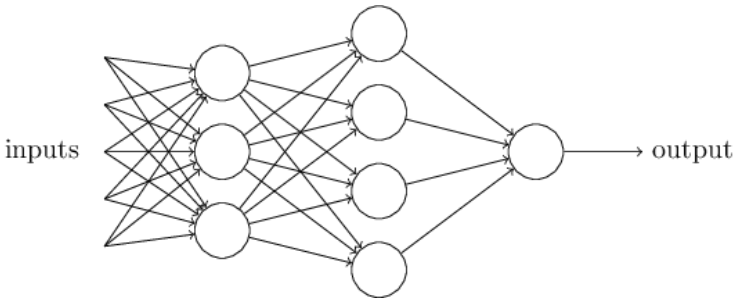
Σχήμα 14: Απεικόνιση εισροών σε ένα Perceptron
By Michael Nielsen - Neural Networks and Deep Learning

Ο Rosenblatt πρότεινε έναν απλό κανόνα για τον υπολογισμό του output. Εισήγαγε τα βάρη, $w_1, w_2, \dots, w_n$, τα οποία είναι πραγματικοί αριθμοί που εκφράζουν τη σημασία των αντίστοιχων εισροών στο output. Το output του νευρώνα, 0 ή 1, καθορίζεται από το εάν το σταθμισμένο άθροισμα $\sum_j w_j x_j$ είναι μικρότερο ή μεγαλύτερο από κάποια τιμή κατωφλίου *(threshold)*. Ακριβώς όπως τα βάρη, το όριο είναι ένας πραγματικός αριθμός που είναι μια παράμετρος του νευρώνα. Δηλαδή με πιο ακριβείς αλγεβρικούς όρους:

$$output = \varphi(v) = \begin{cases} 0 \; if \;\; v = \sum_j w_j x_j \leq threshold \\ 1 \; if \;\; v = \sum_j w_j x_j > threshold \end{cases}$$

Το perceptron λειτουργεί ακριβώς με το παραπάνω απλό μαθηματικό μοντέλο, δηλαδή βάσει της βαρύτητας κάθε στοιχείου που έχει διαθέσιμου και ανάλογα την τιμή του κατωφλιού που έχουμε ορίσει λαμβάνει τις αποφάσεις.

Προφανώς, το perceptron δεν είναι ένα πλήρες μοντέλο ανθρώπινης λήψης αποφάσεων. Αλλά είναι ένα μοντέλο το οποίο μπορεί να ζυγίσει διάφορα είδη αποδεικτικών στοιχείων για να πάρει αποφάσεις. Και θα πρέπει να φαίνεται εύλογο ότι ένα πολύπλοκο δίκτυο perceptrons θα μπορούσε να λάβει πολύ πιο λεπτές αποφάσεις.



Σχήμα 15: Πολύπλοκο Δίκτυο από Perceptrons
By Michael Nielsen - Neural Networks and Deep Learning

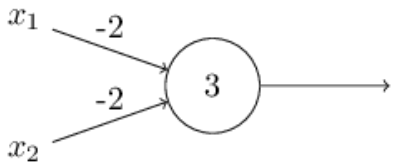
Στο παραπάνω σχήμα, η πρώτη στήλη από perceptrons, την οποία ονομάζουμε ως πρώτο στρώμα *(layer)* από perceptrons, λαμβάνει τρείς απλές αποφάσεις σταθμίζοντας τα αποδεικτικά στοιχεία που του έχουν δοθεί. Αντίστοιχα τα perceptrons του δεύτερου στρώματος, λαμβάνουν αποφάσεις σταθμίζοντας τα αποτελέσματα του πρώτου στρώματος, ώστε ένα perceptron του δεύτερου στρώματος, να μπορεί να πάρει μία απόφαση σε ένα πιο πολύπλοκο επίπεδο από τον πρώτο στρώμα και ακολούθως ακόμα πιο πολύπλοκες αποφάσεις μπορούν να παρθούν από το τρίτο στρώμα. Έτσι με αυτό τον τρόπο ένα TNN με πολλαπλά στρώματα μπορεί να λάβει αποφάσεις με πολύ πιο αποτελεσματικό τρόπο.

Μπορούμε να απλοποιήσουμε τον τρόπο που περιγράφουμε τα perceptrons. Η προϋπόθεση $\sum_j w_j x_j > threshold$, δεν είναι πολύ εύχρηστη, γι' αυτό μπορούμε να κάνουμε δύο αλλαγές για να το απλοποιήσουμε. Αρχικά μπορούμε να θεωρήσουμε το $\sum_j w_j x_j$ ως $\boldsymbol{w} \cdot \boldsymbol{x}$, όπου $\boldsymbol{w}$ και $\boldsymbol{x}$ είναι διανύσματα των οποίων οι συνιστώσες είναι τα βάρη και οι είσοδοι αντίστοιχα. Έπειτα μπορούμε να μετακινήσουμε το κατώφλι στην άλλη πλευρά της ανίσωσης και να το αντικαταστήσουμε με το perceptron's bias[(1)], $b \equiv -threshold$, οπότε τώρα θα προκύπτει ότι

$$output = \begin{cases} 0 \; if \; \boldsymbol{w} \cdot \boldsymbol{x} + b \leq 0 \\ 1 \; if \; \boldsymbol{w} \cdot \boldsymbol{x} + b > 0 \end{cases}$$

Τη πόλωση *(bias)* μπορούμε να τη θεωρήσουμε ως ένα μέτρο του πόσο εύκολου είναι να παράξει το perceptron ως έξοδο την τιμή 1. Για ένα perceptron με μια πολύ μεγάλη πόλωση, είναι εξαιρετικά εύκολο να βγάλει ένα ως έξοδο την τιμή 1. Αλλά αν η πόλωση είναι πολύ αρνητική, τότε είναι δύσκολο για το perceptron να βγάλει τιμή 1.

Μέχρι τώρα έχουμε περιγράψει τα perceptrons ως μια μέθοδο η οποία ζυγίζει τα στοιχεία για τη λήψη αποφάσεων. Ένας άλλος τρόπος όπου τα perceptrons μπορούν να χρησιμοποιηθούν είναι για να υπολογίσουν τις στοιχειώδεις λογικές λειτουργίες όπως οι AND, OR και NAND. Για παράδειγμα, ας υποθέσουμε ότι έχουμε ένα perceptron με δύο εισόδους, το καθένα με βάρος -2 -2, και μια τιμή κατωφλιού 3.



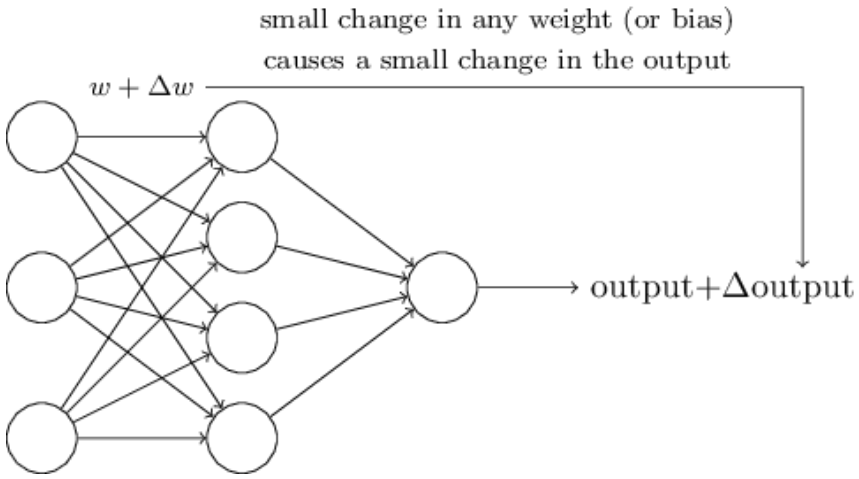
Σχήμα 16

Παρατηρούμε ότι αν θέσουμε ως τιμές εισόδου 0 0, τότε παράγεται ως έξοδο η τιμή 1, αφού $(-2) * 0 + (-2) * 0 + 3 = 3$ είναι θετικός αριθμός. Αντίστοιχα αν θέσουμε ως τιμές εισόδου 1 1, τότε παρατηρούμε ότι ως έξοδο παράγεται η τιμή 0, αφού $(-2) * 1 + (-2) * 1 + 3 = -1$ είναι αρνητικός αριθμός. Με αυτό τον τρόπο το συγκεκριμένο perceptron δημιουργεί μια NAND gate[(2)].

Το παραπάνω παράδειγμα μας δείχνει ότι μπορούμε να χρησιμοποιήσουμε τα perceptrons για να υπολογίσουμε τις απλές λογικές συναρτήσεις. Για την ακρίβεια μπορούμε να τα χρησιμοποιήσουμε για να υπολογίσουμε οποιαδήποτε λογική συνάρτηση. Ο λόγος είναι ότι μέσω μίας NAND gate μπορούμε να κατασκευάσουμε οποιονδήποτε υπολογισμό.

3.2.2 Σιγμοειδής Συνάρτηση

Τα perceptrons ενώ είναι πολύ απλά και εύχρηστα, έχουν όμως ένα πρόβλημα όσον αφορά της τεχνικές εκμάθησης. Ας υποθέσουμε ότι έχουμε ένα δίκτυο perceptrons που θα θέλαμε να χρησιμοποιήσουμε και να το εκπαιδεύσουμε για να λύσουμε κάποιο πρόβλημα. Για να δούμε πώς μπορεί να λειτουργήσει η μάθηση, ας υποθέσουμε ότι κάνουμε μια μικρή αλλαγή σε κάποιο βάρος (ή στη πόλωση) στο δίκτυο. Αυτό που θα θέλαμε είναι αυτή η μικρή αλλαγή βάρους να προκαλέσει μόνο μια μικρή αντίστοιχη αλλαγή στην έξοδο από το δίκτυο. Αυτό που θέλουμε να δούμε, παρουσιάζεται σχηματικά στο *σχήμα 17*.



Σχήμα 17 By Michael Nielsen - Neural Networks and Deep Learning

Εάν αυτό ίσχυε όντως, ότι δηλαδή μία μικρή αλλαγή στα βάρη (ή στη πόλωση), θα προκαλούσε μόνο μία μικρή αντίστοιχα αλλαγή στην έξοδο του δικτύου, τότε θα μπορούσαμε να χρησιμοποιήσουμε το γεγονός αυτό για να τροποποιήσουμε τα βάρη (ή τις πολώσεις) ώστε να κάνουμε το δίκτυο να συμπεριφέρεται περισσότερο με τον τρόπο που θέλουμε. Αλλάζοντας τις τιμές στα βάρη (ή στις πολώσεις) ξανά και ξανά, θα μπορούσαμε να εκπαιδεύσουμε το δίκτυο με αυτόν τον απλό θεωρητικά τρόπο.

Το πρόβλημα όμως έγκειται στο γεγονός ότι όταν το δίκτυο περιέχει perceptrons, τότε μία μικρή αλλαγή στα βάρη (ή στις πολώσεις) οποιουδήποτε μεμονωμένου perceptron, μπορεί να προκαλέσει την έξοδο του δικτύου να αναστραφεί πλήρως. Αυτό καθιστά δύσκολο να δούμε πώς να τροποποιούμε σταδιακά τα βάρη και τις πολώσεις, έτσι ώστε το δίκτυο να πλησιάζει περισσότερο στην επιθυμητή συμπεριφορά.

Μπορούμε να ξεπεράσουμε αυτό το πρόβλημα με την εισαγωγή ενός νέου τύπου τεχνητού νευρώνα που ονομάζεται Σιγμοειδής Νευρώνας. Οι Σιγμοειδείς Νευρώνες είναι παρόμοιοι με τους perceptrons, αλλά τροποποιούνται έτσι ώστε οι μικρές αλλαγές στα βάρη τους και στη πόλωση, να προκαλούν μόνο

μικρή αλλαγή στην έξοδο του δικτύου. Αυτό είναι το κρίσιμο γεγονός που θα επιτρέψει σε ένα δίκτυο Σιγμοειδών Νευρώνων να εκπαιδευτούν.

Θα απεικονίσουμε σιγμοειδείς νευρώνες με τον ίδιο τρόπο που απεικονίσαμε perceptrons, στο *σχήμα 17*.

Ακριβώς όπως και με τα perceptrons, ο Σιγμοειδής Νευρώνας, έχει εισόδους $x_1, x_2, \ldots, x_n$, αλλά πλέον τώρα μπορούμε να δώσουμε τιμές εισόδου, οποιαδήποτε τιμή μεταξύ 0 και 1. Επίσης όπως και τα perceptrons ο Σιγμοειδής Νευρώνας, έχει βάρη $w_1, w_2, \ldots, w_n$ για κάθε είσοδο και μία συνολική πόλωση, b. Οι τιμές εξόδου όμως δεν είναι πλέον αποκλειστικά 0 και 1. Αντίθετα είναι $\sigma(w * x + b)$, όπου το σ ονομάζεται σιγμοειδής συνάρτηση[3] και καθορίζεται από

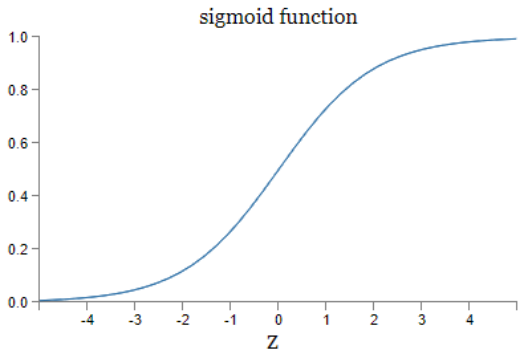
$$\sigma(z) \equiv \frac{1}{1 + e^{-z}}$$

Για να το θέσουμε πιο γενικά, η τιμή εξόδου ενός σιγμοειδούς νευρώνα με εισόδους $x_1, x_2, \ldots, x_n$, βάρη $w_1, w_2, \ldots, w_n$ και πόλωση b είναι

$$\frac{1}{1 + \exp(-\sum_j w_j x_j - b)}$$

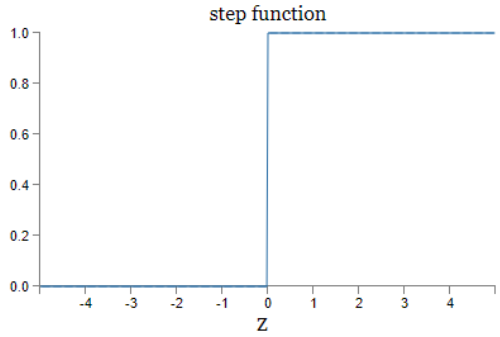
Εκ πρώτης όψεως, οι σιγμοειδείς νευρώνες εμφανίζονται πολύ διαφορετικοί από τους perceptrons. Στην πραγματικότητα, υπάρχουν πολλές ομοιότητες μεταξύ τους. Για να κατανοήσουμε την ομοιότητα αυτή, υποθέτουμε ότι $z \equiv w * x + b$ είναι ένας μεγάλος θετικός αριθμός. Τότε $e^{-z} \approx 0$ και επίσης $\sigma(z) \approx 1$. Με άλλα λόγια, η τιμή εξόδου ενός σιγμοειδή νεύρωνα σε αυτή την περίπτωση είναι προσεγγιστικά 1, όπως θα ήταν και στην περίπτωση ενός perceptron. Αντίστοιχα ισχύει και το ανάποδο, αν δηλαδή $z \equiv w * x + b$ είναι ένας μεγάλος αρνητικός αριθμός, τότε η τιμή εξόδου ενός σιγμοειδή νευρώνα σε αυτή την περίπτωση είναι προσεγγιστικά 0, όπως θα ήταν και στην περίπτωση ενός perceptron.

Όμως, η αλγεβρική μορφή του σ, στην πραγματικότητα δεν είναι τόσο σημαντική, αυτό που είναι πραγματικά σημαντικό είναι η μορφή της συνάρτησης, όπως φαίνεται στο *σχήμα 18*.



Σχήμα 18: Σιγμοείδης Συνάρτηση

η οποία είναι μία εξομαλυμένη μορφή της βηματικής συνάρτησης*(step function)*



Σχήμα 19: Βηματική Συνάρτηση

Στην πραγματικότητα εάν ήταν μία βηματική συνάρτηση, τότε ο σιγμοειδής νεύρωνας θα ήταν ένα perceptron, αφού η τιμή εξόδου θα ήταν 0 ή 1, ανάλογα με το αν το $w * x + b$ ήταν θετικό ή αρνητικό αντίστοιχα.[(4)] Αυτό είναι ένα κρίσιμο συμπέρασμα στο οποίο καταλήγουμε ότι χρησιμοποιώντας την πραγματική σ συνάρτηση παίρνουμε ένα εξομαλυμένο perceptron. Αυτό σημαίνει ότι μικρές αλλαγές στα βάρη και στη πόλωση θα μας δώσουν μικρές αλλαγές και στην τιμή εξόδου του δικτύου.

$$\Delta output \approx \sum_j \frac{\vartheta output}{\vartheta w_j} \Delta w_j + \frac{\vartheta output}{\vartheta b} \Delta b$$

Η παραπάνω έκφραση λέει κάτι απλό και σημαντικό, ότι το $\Delta output$ είναι μία γραμμική συνάρτηση των αλλαγών Δw_j και Δb στα βάρη και στη πόλωση αντίστοιχα. Αυτή η γραμμικότητα, είναι που επιτρέπει μικρές αλλαγές στα βάρη και στη πόλωση να επιφέρουν μικρές αλλαγές στη τιμή εξόδου. Έτσι καταλήγουμε στο συμπέρασμα ότι ενώ οι σιγμοειδής νευρώνες έχουν την ίδια ποιοτική συμπεριφορά με τα perceptrons, παρόλα αυτά καθιστούν πιο εύκολο να κατανοήσουμε πως μικρές αλλαγές στα βάρη ή στη πόλωση επιφέρουν και μικρές αλλαγές στην τιμή εξόδου του δικτύου.

Από τα παραπάνω συμπεραίνουμε ότι προφανώς μία μεγάλη διαφορά μεταξύ των perceptrons και των σιγμοειδών νευρώνων είναι ότι οι τελευταίοι μπορούν και παράγουν τιμές εξόδου οποιονδήποτε πραγματικό αριθμό μεταξύ 0 και 1. Αυτό όμως καμιά φορά μπορεί να μην είναι βολικό. Για παράδειγμα αν θέλουμε να αποφασίσουμε για ένα γεγονός αν ισχύει ή όχι, θα ήταν πιο βολικό το δίκτυο να έδινε τιμές 0 και 1. Αυτό στην πράξη μπορούμε να το προσπεράσουμε δημιουργώντας μία σύμβαση, όπου για παράδειγμα τιμές εξόδου μεγαλύτερες του 0.5 σημαίνει ότι αυτό το γεγονός ισχύει και αντίστοιχα μικρότερες ότι δεν ισχύει.

3.2.3 Η Υπερβολική Εφαπτομένη *(Hyperbolic Tangent)* Συνάρτηση Ενεργοποίησης

Αν και η Σιγμοειδής Συνάρτηση ενεργοποίησης έχει μία ωραία βιολογική ερμηνεία, αυτή μπορεί να προκαλέσει ένα νευρωνικό δίκτυο να κολλήσει κατά τη διάρκεια της εκπαίδευσής του. Αυτό οφείλεται εν μέρει στο γεγονός ότι εάν παρέχεται έντονα αρνητική τιμή εισόδου, εκπέμπει τιμές πολύ κοντά στο μηδέν και αυτό μπορεί να οδηγήσει στο ότι η ενημέρωση των παραμέτρων κατά τη διάρκεια της εκπαίδευσης, να μην γίνεται όσο τακτικά θέλουμε. (Stansbury, 2014)

Έτσι μία ενναλακτική συνάρτηση είναι η Υπερβολική Εφαπτομένη

$$\varphi_{tanh}(\upsilon) = \frac{\sinh(\upsilon)}{\cosh(\upsilon)} = \frac{e^{\upsilon} - e^{-\upsilon}}{e^{\upsilon} + e^{-\upsilon}}$$

Η συνάρτηση αυτή είναι επίσης μία Σιγμοειδής συνάρτηση όσον αφορά το σχήμα της, η διαφορά της όμως βρίσκεται στο γεγονός ότι παίρνει τιμές στο διάστημα $(-1,1)$. Το γεγονός ότι οι τιμές εξόδου είναι κεντραρισμένες στο μηδέν, την κάνει προτιμότερη της Σιγμοειδής Συνάρτησης.

3.2.4 Ανορθωμένη Γραμμική Συνάρτηση Ράμπας *(Rectified Linear Unit – ReLU)*

Η συνάρτηση ράμπας έχει την εξής μορφή

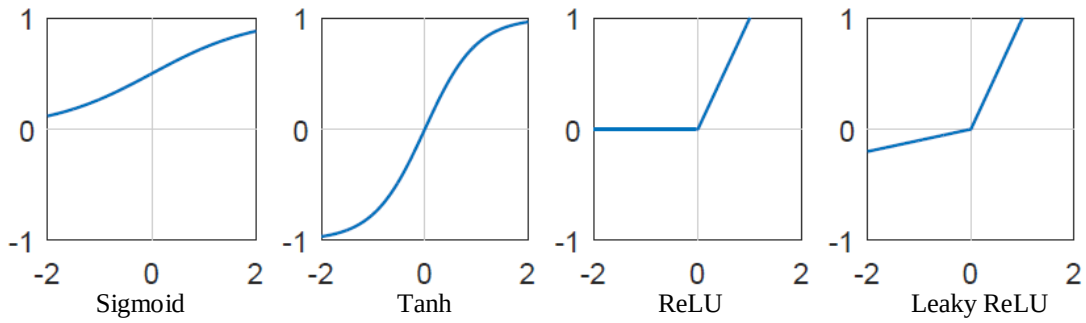
$$\varphi(\upsilon) = \max(0, \upsilon)$$

Η συγκεκριμένη συνάρτηση είναι η πιο δημοφιλής συνάρτηση ενεργοποίησης για Βαθιά Νευρωνικά Δίκτυα (DNN). Προτιμάται από τις υπόλοιπες συναρτήσεις διότι έχει την δυνατότητα να εκπαιδεύσει ένα δίκτυο αρκετά πιο γρήγορα, δίνοντας ακριβή αποτελέσματα. Ωστόσο, ένα σημαντικό μειονέκτημα της είναι πως κάποιες φορές μπορεί να οδηγήσει ορισμένους νευρώνες του δικτύου σε κάποιες τιμές βαρών, οι οποίες τους αποτρέπουν να ενεργοποιηθούν. Έτσι αυτοί οι νευρώνες νεκρώνουν, σταματάνε δηλαδή να εκπαιδεύονται.

Τη λύση στο πρόβλημα αυτό, δίνει η Παραμετροποιημένη Συνάρτηση Ράμπας *(PReLU)*

$$\varphi(\upsilon) = \begin{cases} \upsilon, & \alpha\nu\ \upsilon > 0 \\ \alpha * \upsilon, & \delta\iota\alpha\varphi o\rho\varepsilon\tau\iota\kappa\acute{\alpha} \end{cases}$$

Η συνάρτηση αυτή πολλαπλασιάζει την τιμή εξόδου με μία μικρή τιμή α, στην περίπτωση που η τιμή εισόδου είναι αρνητική. Αν $\alpha = 0$, τότε η συνάρτηση μετατρέπεται σε ReLU, ενώ αν το α, πάρει μία μικρή και σταθερή τιμή ονομάζεται Leaky ReLU.



Σχήμα 20: Οι τέσσερις βασικές συναρτήσεις ενεργοποίησης.

3.2.5 Συνάρτηση Softmax

Όταν θέλουμε να αντιμετωπίσουμε προβλήματα κατηγοριοποίησης, οι προηγούμενες συναρτήσεις, δεν μπορούν να μας βοηθήσουν πολύ. Για παράδειγμα, η σιγμοειδής συνάρτηση, μπορεί να χειριστεί μέχρι δύο κλάσεις, κάτι το οποίο πολύ συχνά δεν μας είναι αρκετό. Η συνάρτηση Softmax, η οποία είναι μία γενίκευση της λογιστικής συνάρτησης, μπορεί να μας βοηθήσει σε αυτό το πρόβλημα. Χρησιμοποιείται συχνά στο τελευταίο στρώμα ενός δικτύου, στο οποίο όπως γνωρίζουμε προκύπτουν οι τιμές εξόδου του δικτύου και λειτουργεί συμπυκνώνοντας τις τιμές αυτές, έτσι ώστε να είναι μεταξύ 0 και 1, αλλά και το άθροισμα τους να ισούται με τη μονάδα. Έτσι κάθε τιμή εξόδου, που προκύπτει από μία συνάρτηση softmax, είναι ισοδύναμη με μία κατηγορική συνάρτηση πιθανότητας. Η συνάρτηση αυτή, χρησιμοποιείται ευρέως στα βαθιά νευρωνικά δίκτυα, στα οποία θα αναφερθούμε στη συνέχεια.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

3.3 Αλγόριθμοι Βελτιστοποίησης Νευρωνικών Δικτύων

Ένα ΤΝΔ, όπως ακριβώς συμβαίνει και με τα βιολογικά ΝΔ, προκειμένου να μπορέσουν να λειτουργήσουν στη βέλτιστη τους μορφή και να λαμβάνουν σωστές αποφάσεις, πρέπει να υποβληθούν σε ένα είδος εκπαίδευσης, η οποία ξεκινά από μία κατάσταση, στην οποία δεν υπάρχει καμία γνώση. Ενώ στη συνέχεια μέσω της εκπαίδευσης, αποκτά την απαραίτητη «εμπειρία», η οποία στα ΤΝΔ αποθηκεύεται στα συνοπτικά βάρη, ώστε να μπορέσει να υλοποιήσει με ακρίβεια το πρόβλημα.

Μπορούμε να κατηγοριοποιήσουμε την διαδικασία εκπαίδευσης σε δύο βασικές κατηγορίες.

1. **Εκπαίδευση με Επίβλεψη *(Supervised Learning).*** Με την μέθοδο αυτή, είναι απαραίτητο να τροφοδοτούνται τα ΤΝΔ με κάποια «πρότυπα», τα οποία αποτελούν και το επιθυμητό αποτέλεσμα του προβλήματος που θα πρέπει να επιλύσουν. Με βάση αυτά το δίκτυο προσαρμόζει τις παραμέτρους του (συνοπτικά βάρη και πολώσεις), μέσω αλγορίθμων, με σκοπό τη σταδιακή μείωση μίας συνάρτησης κόστους που έχει οριστεί. Η διαδικασία αυτή επαναλαμβάνεται μέχρις ότου, το κόστος να μηδενιστεί ή να θεωρηθεί αποδεκτό. Με αυτό το είδος εκπαίδευσης θα ασχοληθούμε και στην παρούσα εργασία εκτενέστερα.
2. **Εκπαίδευση χωρίς Επίβλεψη *(Unsupervised Learning).*** Με την μέθοδο αυτή, δεν υπάρχουν κάποια «πρότυπα» της λειτουργίας που πρέπει να μάθει το δίκτυο, αντιθέτως βασίζονται σε τοπική πληροφορία, οργανώνοντας τα δεδομένα και ανακαλύπτοντας βασικές συλλογικές ιδιότητες. Τα ΤΝΔ με αυτή τη μέθοδο, συνήθως χρησιμοποιούνται για κατάτμηση των δεδομένων(clustering) ή την εκτίμηση της πυκνότητάς τους(density estimation).

3.3.1 Αλγόριθμος Κατάβασης Δυναμικού *(Gradient Descent)*

Όπως αναφέραμε, με τον όρο βελτιστοποίηση, αναφερόμαστε συνήθως στη διαδικασία είτε ελαχιστοποίησης είτε μεγιστοποίησης μίας συνάρτησης $C(x)$, μεταβάλλοντας το x. Στην περίπτωση που μελετάμε, προσπαθούμε να ελαχιστοποιήσουμε μία συνάρτηση, την οποία ονομάζουμε συνάρτηση κόστους *(cost function).*

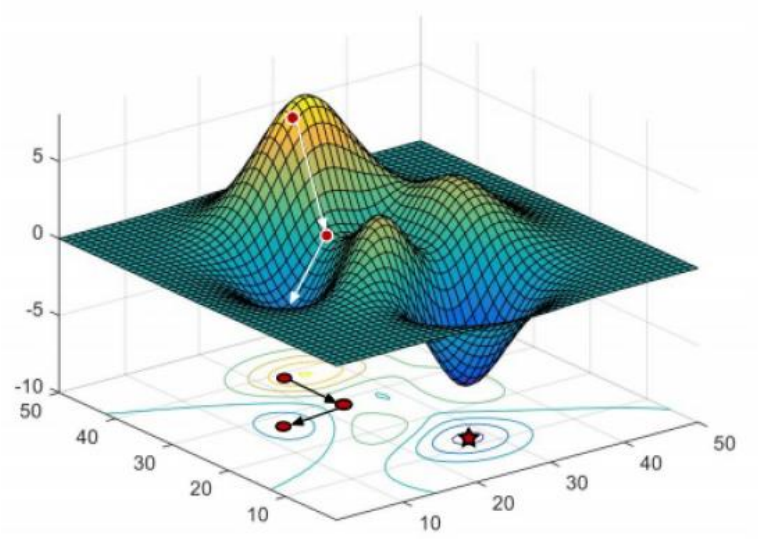
Για να κατανοήσουμε πως μπορούμε να επιτύχουμε τον σκοπό μας αυτό θα ορίσουμε την παρακάτω συνάρτηση κόστους $C(w, b)$.

$$C(w,b) = \frac{1}{2n}\sum_{x}\|y(x) - \boldsymbol{a}\|^2$$

Όπου, w είναι η συλλογή όλων των βαρών στο δίκτυο και b όλων των πολώσεων. Επίσης, n είναι ο συνολικός αριθμός των εισόδων εκπαίδευσης και a, είναι το διάνυσμα των τιμών εξόδων του δικτύου στην x τιμή εισόδου.

Τη συνάρτηση C, την ονομάζουμε και τετραγωνική (*quadratic*) συνάρτηση κόστους, γνωστή και ως συνάρτηση ελαχίστων τετραγώνων MSE. Σκοπός του αλγορίθμου εκπαίδευσης είναι να βρει όλα τα βάρη και τις πολώσεις, ώστε $C(w, b) \approx 0$. Ο αλγόριθμος με τον οποίο πραγματοποιείται η διαδικασία αυτή ονομάζεται Μέθοδος Κατάβασης Δυναμικού (*Gradient Descent*).

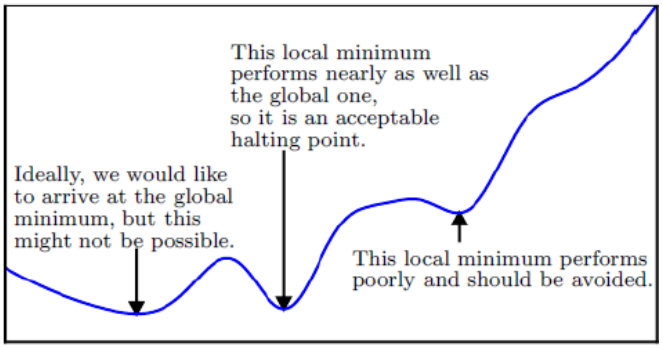
Ας υποθέσουμε ότι προσπαθούμε να ελαχιστοποιήσουμε τη συνάρτηση $C(\boldsymbol{v})$, όπου $v = v_1, v_2, \dots v_n$. Ας θεωρήσουμε, ότι στην περίπτωση μας έχουμε δύο μόνο μεταβλητές $v_{1,}v_2$.



Σχήμα 21: Η μέθοδος κατάβασης δυναμικού για μία συνάρτηση δύο διαστάσεων.

Σκοπός μας, όπως αναφέραμε προηγουμένως, είναι να βρούμε το σημείο, στο οποίο η συνάρτηση κόστους C, επιτυγχάνει το ολικό ελάχιστο (*Global Minimum*). Ένας απλός τρόπος για να λύσουμε το συγκεκριμένο πρόβλημα, είναι ο αναλυτικός. Δηλαδή, να υπολογίσουμε τις μερικές παραγώγους και να προσπαθήσουμε να βρούμε το σημείο στο οποίο το C είναι ακρότατο. Στην περίπτωση των δύο μεταβλητών αυτό, ίσως είναι εύκολο να επιτευχθεί, όμως τα προβλήματα με τα οποία θα ασχοληθούμε έχουν περισσότερες των δύο μεταβλητών και εκεί θα πρέπει να χρησιμοποιήσουμε αλγόριθμο για την επίτευξη του στόχου μας.

Υπάρχουν περιπτώσεις, στις οποίες υπάρχουν περισσότερα από ένα ολικά ελάχιστα και πολλά τοπικά ελάχιστα και συναντούμε σχεδόν αποκλειστικά τέτοιες περιπτώσεις στα πλαίσια του *Deep Learning*. Σε αυτές τις περιπτώσεις πολλές φορές είμαστε ικανοποιημένοι και με τιμές της συνάρτησης κόστους αρκετά μικρές και όχι αναγκαστικά τα ολικά ελάχιστα.



Σχήμα 22: Οι αλγόριθμοι βελτιστοποίησης καμία φορά αποτυγχάνουν να βρουν το ολικό ελάχιστο, όταν υπάρχουν πολλαπλά τοπικά ελάχιστα.
(Goodfellow, Bengio, & Courville, 2016)

Η μέθοδος κατάβασης δυναμικού, ορίζει ως αναγκαία συνθήκη για το βέλτιστο v^* την

$$\nabla C(v^*) = 0$$ [(5)]

Και ισχύει ότι

$$\Delta C(\boldsymbol{v}) \approx \nabla \mathrm{C}(\boldsymbol{v}) * \Delta \boldsymbol{v}^{(6)}$$

Ο αλγόριθμος κατάβασης δυναμικού, ξεκινάει από ένα τυχαίο σημείο και προχωράει με διαδοχικές επισκέψεις σε άλλα σημεία, τέτοια ώστε η συνάρτηση κόστους $C(\boldsymbol{v})$, να μειώνεται σε κάθε επανάληψη, σύμφωνα με τη σχέση

$$C\big(v(k+1)\big) < C(v(k))$$

Όπου $v(k)$, η τιμή του διανύσματος κατά το βήμα k και αντίστοιχα $v(k+1)$, η τιμή του διανύσματος κατά το βήμα $k+1$. Οι διαδοχικές τιμές του διανύσματος v, θα πρέπει να είναι προς την κατεύθυνση της πλέον απότομης κατάβασης, επομένως σε αντίθετη κατεύθυνση προς το διάνυσμα κλίσης $\nabla C(\mathbf{v})$. Ο κανόνας κατάβασης δυναμικού θα είναι

$$v \to v' = v - \eta \nabla C$$

όπου η, είναι ο ρυθμός εκμάθησης.

Για να κάνουμε τον αλγόριθμο της κατάβασης δυναμικού να λειτουργεί σωστά, θα πρέπει από τη μία πλευρά να διαλέξουμε το η αρκετά μικρό, ώστε η παραπάνω εξίσωση να έχει σωστή προσέγγιση και από την άλλη πλευρά, όχι τόσο μικρό, διότι ο αλγόριθμος θα λειτουργεί με πάρα πολύ αργό ρυθμό.

Χρησιμοποιώντας τον κανόνα αυτόν ξανά και ξανά, θα ελαχιστοποιούμε σε κάθε βήμα το C, μέχρι να φθάσουμε στο ολικό ελάχιστο.

Η παραπάνω διαδικασία, ισχύει και στην περίπτωση την οποία έχουμε περισσότερες των δύο μεταβλητών. Στην πράξη η μέθοδος αυτή λειτουργεί με πολύ καλά αποτελέσματα και βοηθάει την μάθηση του δικτύου.

Όσον αφορά το μοντέλο των νευρωνικών δικτύων, η χρήση της μεθόδου κατάβασης δυναμικού, όπως είχαμε αναφέρει και προηγουμένως, χρησιμοποιείται στο να βρεθούν τα βάρη w_k και οι πολώσεις b_i, ώστε να ελαχιστοποιηθεί η συνάρτηση κόστους

$$C(w,b) = \frac{1}{2n}\sum_x \|y(x) - \boldsymbol{a}\|^2$$

Ο κανόνας κατάβασης δυναμικού πλέον θα είναι

$$w_k \to {w_k}' = w_k - \eta \frac{\vartheta C}{\vartheta w_k}$$

$$b_i \to {b_i}' = b_i - \eta \frac{\vartheta C}{\vartheta b_i}$$

Επαναλαμβάνοντας συνέχεια αυτόν τον κανόνα, μπορούμε να βρούμε το ολικό ελάχιστο της συνάρτησης μας, δηλαδή με άλλα λόγια, αυτός είναι ο κανόνας εκμάθησης για ένα νευρωνικό δίκτυο.

Στην πράξη όμως υπάρχουν αρκετές δυσκολίες στην εφαρμογή του παραπάνω κανόνα. Μία από αυτές είναι ότι κοιτώντας τη συνάρτηση κόστους $C(w,b)$, παρατηρούμε ότι έχει τη μορφή $C = \frac{1}{n}\sum C_x$, η οποία είναι μέσος όρος των κοστών $C_x = \frac{\|y(x)-\boldsymbol{a}\|^2}{2}$. Αυτό στην πραγματικότητα σημαίνει ότι θα πρέπει να υπολογίσουμε τη κλίση της συνάρτησης κόστους ∇C_x για κάθε ξεχωριστή τιμή εισόδου εκπαίδευσης

και στη συνέχεια να πάρουμε τον μέσο όρο τους. Όμως αυτό όταν έχουμε πάρα πολλές τιμές εισόδου εκπαίδευσης στο δίκτυο μας, οι υπολογισμοί αυτοί μπορούν να διαρκέσουν υπερβολικά μεγάλο χρόνο.

Μία ιδέα για να επιταχύνουμε την εκμάθηση, είναι η μέθοδος της στοχαστικής κατάβασης δυναμικού *(stochastic gradient descent)*. Η μέθοδος αυτή χρησιμοποιεί ένα υποσύνολο του συνόλου εκπαίδευσης, προκειμένου να υπολογίσει την κλίση της συνάρτησης κόστους ∇C. Ο αριθμός των δειγμάτων εκπαίδευσης που χρησιμοποιούνται σε μία επανάληψη του αλγορίθμου ονομάζεται μέγεθος παρτίδας *(batch size)*, η επιλογή της οποίας είναι τυχαία. Εδώ, ο όρος "στοχαστικός" προέρχεται από το γεγονός ότι η κλίση (*gradient)* που βασίζεται σε ένα δείγμα εκπαίδευσης είναι μια "στοχαστική προσέγγιση" της "πραγματικής" κλίσης κόστους. Ο στοχαστικός αυτός χαρακτήρας της μεθόδου, έχει το επιθυμητό αποτέλεσμα ότι μειώνει την πιθανότητα η διαδικασία μάθησης να παγιδευτεί σε ένα τοπικό ελάχιστο.

Υπάρχουν και άλλες παραλλαγές της Διαδικασίας Κατάβασης Δυναμικού, όμως για τις ανάγκες τις εργασίας, θα αναφθούμε σε αυτούς ονομαστικά. Από αυτούς οι Adam και RMSprop, είναι αυτοί που χρησιμοποιούνται περισσότερο στην πράξη σήμερα.

- Adam
- RMSprop
- Adagrad
- Adadelta
- Adamax
- AMSGrad
- Momentum
- Nesterov Momentum
- Nadam
- AMSGrad

3.4 Αλγόριθμος Back Propagation *(BP)*

Στο προηγούμενο κομμάτι, αναφερθήκαμε στο πως τα μοντέλα νευρωνικών δικτύων, μπορούν να εκπαιδευτούν μετατρέποντας τα βάρη τους. Δεν αναφερθήκαμε όμως στη διαδικασία υπολογισμού της κλίσης(gradient) της συνάρτησης κόστους, η διαδικασία αυτή επιτυγχάνεται μέσω του αλγόριθμου Back Propagation.

Ο αλγόριθμος αυτός, αναπτύχθηκε αρχικά στην δεκαετία του 70', όμως η σημαντικότητα του, άργησε να εκτιμηθεί, έως ότου το 1984 οι David Rumelhart, Geoffrey Hinton, και Ronald Williams, δημοσίευσαν ένα paper, το οποίο περιέγραφε πως ο αλγόριθμος αυτός λειτουργούσε πολύ πιο γρήγορα σε σχέση με τους προηγούμενους και έδινε λύσεις σε προβλήματα μέχρι τότε άλυτα. Επίσης μέσω του αλγορίθμου αυτού, παίρνουμε λεπτομερείς πληροφορίες για το πως αλλάζοντας τιμές στα βάρη *(ή τις πολώσεις)*, αλλάζει η ολική συμπεριφορά του δικτύου.

Ο αλγόριθμος BP, περιλαμβάνει δύο διαφορετικές φάσεις:

1. Στην πρώτη φάση, το σήμα εισόδου μεταδίδεται στο δίκτυο προς τα εμπρός, μέχρι να φθάσει στο τελευταίο επίπεδο εξόδου και τα συνοπτικά βάρη του δικτύου παραμένουν σταθερά.
2. Στην δεύτερη φάση, η οποία εξελίσσεται προς τα πίσω, ένα σήμα σφάλματος παράγεται ως η διαφορά της πραγματικής εξόδου του δικτύου με μία επιθυμητή έξοδο και επίπεδο προς επίπεδο το σήμα αυτό διαδίδεται προς τα πίσω, προσαρμόζοντας τα συνοπτικά βάρη του δικτύου.

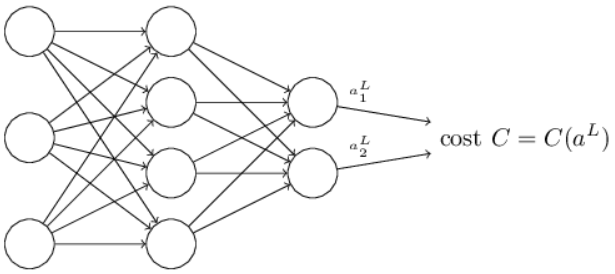
Σκοπός του BP είναι να υπολογιστούν οι μερικές παράγωγοι $\frac{\partial C}{\partial w}$ $και$ $\frac{\partial C}{\partial b}$, της συνάρτησης κόστους C με καθένα από τα βάρη και τις πολώσεις του δικτύου. Θα χρησιμοποιήσουμε την τετραγωνική συνάρτηση κόστους, η οποία σύμφωνα και με αυτό που μόλις αναφέραμε αυτή θα έχει τη μορφή

$$C = \frac{1}{2n}\sum_x \|y(x) - a^L(x)\|^2$$

Όπου το L αναφέρεται στον αριθμό των επιπέδων του δικτύου και $a^L(x)$ είναι το διάνυσμα των εξόδων ενεργοποίησης στην x τιμή εισόδου στο δίκτυο.

Στη συνέχεια, θα πρέπει να κάνουμε δύο υποθέσεις όσον αφορά τη συνάρτηση κόστους.

1. Την πρώτη την είχαμε αναφέρει και σε προηγούμενη ενότητα και αφορά το γεγονός ότι η συνάρτηση κόστους μπορεί να γραφεί σαν μέσος όρος $C = \frac{1}{n}\sum C_x$ συναρτήσεων κόστους C_x. Ο λόγος που χρειαζόμαστε την υπόθεση αυτή είναι επειδή μέσω του BP υπολογίζουμε τις μερικές παραγώγους $\frac{\partial C_x}{\partial w}$ $και$ $\frac{\partial C_x}{\partial b}$, για κάθε ένα παράδειγμα εκπαίδευσης, μπορούμε να υπολογίσουμε τις μερικές παραγώγους $\frac{\partial C}{\partial w}$ $και$ $\frac{\partial C}{\partial b}$ παίρνοντας τον μέσο όρο των παραδειγμάτων εκπαίδευσης.
2. Η δεύτερη υπόθεση που κάνουμε για τη συνάρτηση κόστους είναι ότι μπορεί να γραφτεί ως συνάρτηση των εξόδων από το νευρωνικό δίκτυο.



Σχήμα 23: Δεύτερη υπόθεση συνάρτησης κόστους.
By Michael Nielsen - Neural Networks and Deep Learning

Η τετραγωνική συνάρτηση κόστους, ικανοποιεί αυτή την υπόθεση, αφού για ένα απλό παράδειγμα εκπαίδευσης x, αυτή μπορεί να γραφεί στη μορφή

$$C = \frac{1}{2}\|y(x) - a^L(x)\|^2 = \frac{1}{2}\sum_j (y_j - a_j^L)^2$$

η οποία εξαρτάται από την επιθυμητή τιμή εξόδου y.

Ο αλγόριθμος BP, αφορά να καταλάβουμε το πως αλλάζοντας τα βάρη και τις πολώσεις σε ένα δίκτυο, αλλάζει η συνάρτηση κόστους. Αυτό σημαίνει ότι πρέπει να υπολογίσουμε τις μερικές παραγώγους $\frac{\partial C_x}{\partial w_{jk}^l}$ $και$ $\frac{\partial C_x}{\partial b_j^l}$.

Όπου w_{jk}^l αναφέρεται στο βάρος από τον k-οστό νευρώνα στο $(l-1)$ στρώμα, προς τον j-οστό νευρώνα του l-οστού στρώματος. Αντίστοιχα και το b_j^l αναφέρεται στη πόλωση του j-οστού νευρώνα του l-οστού στρώματος.

Πριν υπολογίσουμε τις προαναφερθείσες μερικές παραγώγους, πρέπει να εισάγουμε την ποσότητα

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}$$

την οποία καλούμε σφάλμα στον j-οστό νευρώνα του l-οστού στρώματος. Όπου z_j^l είναι η τιμή ενεργοποίησης του νευρώνα j του l-οστού στρώματος.

Ο αλγόριθμος BP, μας δίνει τη δυνατότητα να υπολογίσουμε το διάνυσμα σφαλμάτων δ^l κάθε στρώματος[(7)] και στη συνέχεια να σχετίσουμε αυτά τα σφάλματα με τις μερικές παραγώγους που μας ενδιαφέρουν.

Στη συνέχεια θα εισάγουμε τέσσερις βασικές εξισώσεις στις οποίες βασίζεται ο αλγόριθμος ΒΚ:

1. Εξίσωση σφάλματος ενός στρώματος, δ^L

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

Ο πρώτος όρος στα δεξιά υπολογίζει το πόσο γρήγορα αλλάζει η συνάρτηση στη j-οστή έξοδο ενεργοποίησης. Ο δεύτερος όρος, υπολογίζει το πόσο γρήγορα η συνάρτηση ενεργοποίησης σ, αλλάζει στην τιμή ενεργοποίησης z_j^L. Η παραπάνω εξίσωση είναι μία πολύ καλή έκφραση, αλλά για τον αλγόριθμο BP, χρειαζόμαστε μία έκφραση βασισμένη σε πίνακες, αυτή είναι

$$\delta^L = \nabla_\alpha C \odot \sigma'(z^L) \text{ [(8)]} \qquad \textbf{(BP1)}$$

2. Εξίσωση σφάλματος δ^l με το σφάλμα δ^{l+1}του επόμενου στρώματος.

$$\delta^l = \left(\left(w^{l+1}\right)^T \delta^{l+1}\right) \odot \sigma'(z^l) \qquad \textbf{(BP2)}$$

Όταν εφαρμόζουμε τον ανάστροφο πίνακα $\left(w^{l+1}\right)^T$, μπορούμε να το σκεφτούμε διαισθητικά σαν να μετακινούμε το σφάλμα προς τα πίσω μέσω του δικτύου, δίνοντας μας ένα είδος μέτρησης του σφάλματος του l-οστού στρώματος. Έπειτα παίρνουμε το *Hadamard* γινόμενο $\odot\sigma'(z^l)$. Αυτές οι οπισθοδρομικές κινήσεις του σφάλματος μέσω της συνάρτησης ενεργοποίησης του l-οστού στρώματος, μας δίνουν το σφάλμα στην σταθμισμένη είσοδο του στρώματος l. Συνδυάζοντας τις *(BP1)* και *(BP2)*, μπορούμε να υπολογίσουμε το σφάλμα δ^l για κάθε στρώμα στο δίκτυο.

3. Εξίσωση για τον ρυθμό μεταβολής του κόστους σε σχέση με οποιαδήποτε πόλωση στο δίκτυο.

$$\frac{\vartheta C}{\vartheta b_j^l} = \delta_j^l \qquad \textbf{(BP3)}$$

4. Εξίσωση για τον ρυθμό μεταβολής του κόστους σε σχέση με οποιοδήποτε βάρος στο δίκτυο.

$$\frac{\vartheta C}{\vartheta w_{jk}^l} = a_k^{l-1}\delta_j^l \qquad \textbf{(BP4)}$$

Μία συνέπεια που προκύπτει μέσω της παραπάνω εξίσωσης, είναι ότι τα βάρη εξόδου από νευρώνες χαμηλής ενεργοποίησης, εκπαιδεύονται αργά.

Συνοψίζοντας ο αλγόριθμος BP, παίρνει την ακόλουθη τελική μορφή.

Αλγόριθμος: Back – Propagation

1. **Είσοδος x:** Όρισε την αντίστοιχη ενεργοποίηση a για το στρώμα εισόδου.

2. **Εμπρόσθια Τροφοδότηση:** Για κάθε $l = 2,3, \dots L$ υπολόγισε τα $z^l = w^l a^{l-1} + b^l$ και τα $a^l = \sigma(z^l)$.

3. **Σφάλμα Εξόδου δ^l:** Υπολόγισε το διάνυσμα $\delta^L = \nabla_a C \odot \sigma'(z^L)$.

4. **Οπισθοδρόμησε(*Backpropagate*) το σφάλμα:** Για κάθε $l = L-1, L-2, \dots, 2$ υπολόγισε το $\delta^l = \left(\left(w^{l+1} \right)^T \delta^{l+1} \right) \odot \sigma'(z^l)$

5. **Έξοδος:** Η κλίση της συνάρτησης κόστους δίνεται από

$$\frac{\vartheta C}{\vartheta w_{jk}^l} = a_k^{l-1} \delta_j^l \quad \text{και} \frac{\vartheta C}{\vartheta b_j^l} = \delta_j^l$$

3.5 Άλλες Συναρτήσεις Κόστους

Μέχρι τώρα, έχουμε αναφέρει σαν συνάρτηση κόστους μόνο το μέσο τετραγωνικό σφάλμα *(MSE)*. Υπάρχουν όμως και άλλες συναρτήσεις, οι οποίες χρησιμοποιούνται ευρέως στην πράξη.

Μία συνάρτηση για να θεωρηθεί συνάρτηση κόστους, πρέπει να ικανοποιεί τρείς υποθέσεις:

- Η πρώτη είναι ότι θα πρέπει να εκφράζεται σαν μέσος όρος
$$\mathrm{C} = \frac{1}{n} \sum_x C_x$$
- Η δεύτερη αφορά το γεγονός ότι μία συνάρτηση κόστους, θα πρέπει να είναι ανεξάρτητη από τιμές ενεργοποίησης ενός δικτύου, εκτός από τις τιμές εξόδου a^L.
- Τέλος, μία συνάρτηση κόστους, πρέπει να παίρνει τιμές μεταξύ $0 \leq C \leq 1$.

Η βασικότερη στην οποία θα αναφερθούμε είναι το κόστος δεντροπίας *(cross-entropy)*. Με τη συνάρτηση αυτή, μπορούμε να βελτιώσουμε την ταχύτητα εκμάθησης του αλγορίθμου, κυρίως όταν οι αρχικές τιμές των παραμέτρων είναι αρκετά μακριά από τις βέλτιστες που ψάχνουμε.

$$C = -\frac{1}{n} \sum_x [y * lna + (1-y) * \ln(1-a)]$$

Μία άλλη επιλογή, είναι η συνάρτηση κόστους SVM. Η συνάρτηση αυτή, απευθύνεται κυρίως στο πρόβλημα της ταξινόμησης και έχει τη μορφή

$$C = \frac{1}{n} \sum_x \max\big(0, 1 - y(w * x - b)\big) + \lambda \|w\|^2$$

ΚΕΦΑΛΑΙΟ 4

LSTM Recurrent Neural Networks

Στο προηγούμενο κεφάλαιο αναπτύξαμε την λογική πάνω στην οποία λειτουργούν τα νευρωνικά δίκτυα γενικότερα. Σε αυτό το κεφάλαιο θα ασχοληθούμε με το είδος των νευρωνικών που θα μας απασχολήσουν για την εφαρμογή της παρούσας εργασίας και είναι τα LSTM recurrent neural networks.

4.1 Εισαγωγή

Τα RNN αποτελούν ένα πολύ ισχυρό είδος νευρωνικών δικτύων και ανήκουν στους πιο ελπιδοφόρους αλγορίθμους αυτή τη χρονική περίοδο, επειδή είναι οι μόνοι με εσωτερική μνήμη και συνδυάζουν δύο ιδιότητες.

1. Αποτελούνται από την κρυφή κατάσταση, όπου μπορούν να αποθηκεύουν μεγάλο μέρος πληροφόρησης από το παρελθόν αποτελεσματικά.
2. Έχουν μη γραμμική δυναμική, η οποία τους επιτρέπει να ενημερώνουν την κρυφή τους κατάσταση με πολλούς τρόπους.

Τα RNN είναι σχετικά παλιά, όπως πολλοί άλλοι αλγόριθμοι βαθιάς μάθησης. Αρχικά δημιουργήθηκαν στη δεκαετία του '80, αλλά μπόρεσαν να δείξουν τις πραγματικές τους δυνατότητες μόλις πριν από μερικά χρόνια, λόγω της υπολογιστικής ισχύος, όπως έχουμε αναφέρει. Ειδικότερα με την εφεύρεση των LSTM την δεκαετία του ’90, κατάφεραν να φέρουν επανάσταση στον τομέα αυτόν. Η ιδέα πίσω από τη δημιουργία τους, ξεκίνησε από το γεγονός ότι στα κλασικά νευρωνικά δίκτυα, κάνουμε την υπόθεση ότι όλες οι τιμές εισόδου και εξόδου στο μοντέλο είναι ανεξάρτητες μεταξύ τους, αλλά για πολλές εφαρμογές αυτό δεν είναι καθόλου χρήσιμο, όπως για παράδειγμα στην περίπτωση που θέλουμε να προβλέψουμε την επόμενη λέξη σε ένα κείμενο, χρειάζεται να γνωρίζουμε ποιες είχαν προηγηθεί. Έτσι, λόγω της δομής τους που αποτελείται από εσωτερική μνήμη, τα RNN είναι σε θέση να θυμούνται σημαντικά πράγματα σχετικά με την εισροή που έλαβαν, γεγονός που τους επιτρέπει να είναι πολύ ακριβή στην πρόβλεψη. Αυτός είναι ο λόγος για τον οποίο είναι ο προτιμώμενος αλγόριθμος για ακολουθιακά δεδομένα όπως χρονολογικές σειρές, ομιλία, κείμενο, οικονομικά δεδομένα, ήχος, βίντεο, καιρικές συνθήκες και πολλά άλλα, επειδή μπορούν να διαμορφώσουν μια βαθύτερη κατανόηση μιας ακολουθίας και του πλαισίου της, σε σύγκριση με άλλους αλγορίθμους.

4.2 Τρόπος Λειτουργίας

Σε αντίθεση με τα Feed – Forward Νευρωνικά Δίκτυα, στα οποία η πληροφορία που εισέρχεται κατευθύνεται προς μία μόνο κατεύθυνση, με αποτέλεσμα να μην περνάει από έναν κόμβο δεύτερη φορά, στα RNN, η πληροφορία κάνει κύκλο στο δίκτυο μέσω ενός βρόχου με αποτέλεσμα, όταν λαμβάνει μία απόφαση, μπορεί και λαμβάνει υπόψιν πέραν της τρέχουσας εισροής και την πληροφορία που έχει

συγκρατήσει από προηγούμενες. Ένα συνηθισμένο RNN έχει βραχυπρόθεσμη μνήμη, τα LSTM, τα οποία θα μελετήσουμε εκτενέστερα παρακάτω, έχουν επίσης και μακροπρόθεσμη μνήμη. (Olah, 2015)

Επομένως, τα RNN έχουν δύο εισόδους, μία που αφορά την πρόσφατη πληροφορία και μία την παρελθοντική, οι οποίες συνδυάζονται προκειμένου να καθορίσουν πως θα αντιδράσουν σε νέα εισροή δεδομένων, κάτι το οποίο είναι πιο κοντά στο πως ο ανθρώπινος εγκέφαλος λαμβάνει αποφάσεις μέσω της προηγούμενης εμπειρίας του.

Η πληροφορία διατηρείται σε ένα RNN στην κρυφή κατάσταση του και έτσι μπορεί και επηρεάζει την επεξεργασία κάθε νέου εισερχόμενου δεδομένου. Αυτή βρίσκει εξαρτήσεις μεταξύ γεγονότων σε διαδοχικές χρονικές στιγμές, οι οποίες ονομάζονται «Μακροχρόνιες Εξαρτήσεις» *(long term dependencies)*.

Μπορούμε να περιγράψουμε την διαδικασία αυτή μαθηματικά ως εξής

$$h_t = \Phi(Wx_t + Uh_{t-1})$$

Όπου h_t, είναι η κρυμμένη κατάσταση τη χρονική στιγμή t και αποτελεί μία συνάρτηση του αθροίσματος των τιμών εισόδου την ίδια χρονική περίοδο x_t, πολλαπλασιασμένη από τον πίνακα βαρών W και της κρυμμένης κατάστασης τη χρονική στιγμή $t-1$, πολλαπλασιασμένη από έναν πίνακα μετάβασης U.

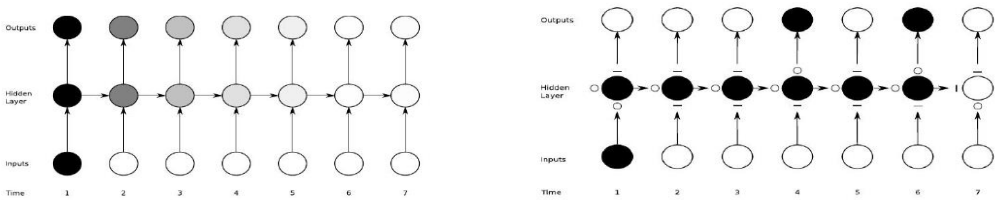
Οι πίνακες με τα βάρη, αποτελούν φίλτρα που καθορίζουν πόση σημασία πρέπει να δοθεί τόσο στην τρέχουσα είσοδο όσο και στην προηγούμενη κρυφή κατάσταση. Το σφάλμα που παράγουν θα επιστρέψει μέσω του αλγόριθμου backpropagation και θα χρησιμοποιηθεί για να προσαρμόσει τα βάρη τους.

Η παραπάνω διαδικασία είναι παρόμοια με μία Μαρκοβιανή αλυσίδα *(Markov chain)*[(9)].

4.3 LSTM

Αν και στη θεωρία, τα κλασικά RNNs φαίνεται να είναι ικανά να διαχειριστούν «Μακροχρόνιες Εξαρτήσεις», στην πράξη όμως, εμφανίζουν προβλήματα στο να τις διαχειριστούν και να μάθουν να ενώνουν αποτελεσματικά την παρελθοντική πληροφόρηση με τη νέα για μεγάλες χρονικές περιόδους. Στη βιβλιογραφία αυτό αναφέρεται ως «vanishing gradient problem» (*Σχήμα 24)*. Σε αυτό το πρόβλημα ήρθαν να δώσουν τη λύση τα LSTM Recurrent Neural Networks, τα οποία εισήχθησαν για πρώτη φορά από τους Hochreiter & Schmidhuber το 1997.

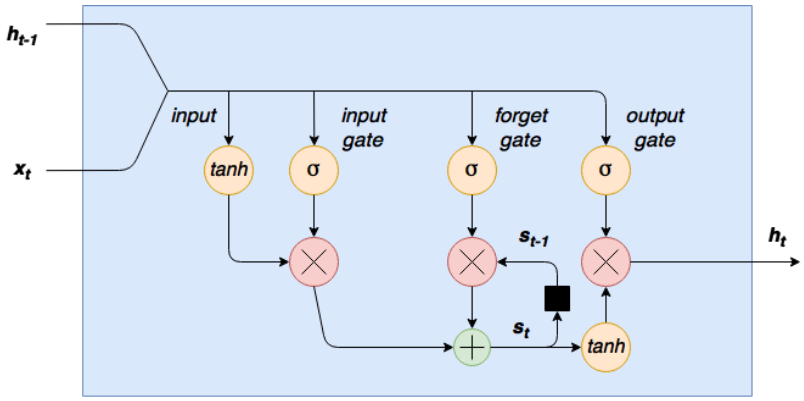
Το «vanishing gradient problem», αναφέρεται στο πρόβλημα εκμάθησης νευρωνικών δικτύων με πολλά κρυμμένα στρώματα, όπου όταν εκτελούμε την μέθοδο BackPropagation, ο υπολογισμός του σφάλματος γίνεται δυσκολότερος με αποτέλεσμα οι αλλαγές στα βάρη να μην γίνονται αποτελεσματικά στην φάση της οπισθοδρόμησης και στην χειρότερη των περιπτώσεων να σταματάει τελείως η εκμάθηση του δικτύου. Το πρόβλημα αυτό αντιμετωπίζουν τα LSTM, με τα μπλοκ «μνήμης» τα οποία βοηθάνε στη διατήρηση της πληροφορίας στο δίκτυο.



Σχήμα 24: Σύγκριση Vanilla RNN(αριστερά) και LSTM(δεξιά) όσον αφορά το "vanishing gradient problem". By Tingwu Wang, University of Toronto

4.3.1 Αρχιτεκτονική LSTM

Η αρχιτεκτονική των LSTM, δεν διαφέρει πολύ από τα κλασικά RNN, παρά μόνο στο ότι χρησιμοποιούν διαφορετική συνάρτηση για να υπολογίσουν την κρυμμένη κατάσταση. Η αρχιτεκτονική τους, αποτελείται από ένα σύνολο επαναλαμβανόμενων *(recurrent)* συνδεδεμένων υπό-δικτύων, γνωστά ως μπλοκ μνήμης. Κάθε τέτοιο μπλοκ περιέχει ένα ή περισσότερα αυτό-συνδεδεμένα κελιά και τρεις πύλες, την πύλη εισόδου, την πύλη εξόδου και την forget πύλη, οι οποίες αποτελούνται από σιγμοειδή συνάρτηση που λαμβάνει τιμές από 0 έως 1. Η τιμή που θα λάβει καθορίζει το ποσοστό της πληροφορίας που θα περάσει για να προστεθεί στη μνήμη, με 0 να σημαίνει ότι δε θα επιτρέψει σε τίποτα να περάσει και με 1 η πληροφορία θα προσχωρήσει αυτούσια. (Thomas, 2018)



Σχήμα 25: LSTM Μπλοκ Μνήμης.
(Thomas, 2018)

Μία πύλη εισόδου, είναι ένα στρώμα από σιγμοειδείς κόμβους, με τιμές εξόδου 0 ή 1, οι οποίοι μπορούν να διώξουν όποια στοιχεία του διανύσματος εισόδου θεωρούν αχρείαστα. Η πύλη forget, ελέγχει το ποια πληροφορία από την προηγούμενη κατάσταση θα περάσει στην τωρινή ή θα ξεχαστεί. Οι δύο αυτές πύλες μαζί, επιτρέπουν στο δίκτυο να ελέγχει ποια πληροφορία θα αποθηκεύεται και ποια θα αντικατασταθεί σε κάθε βήμα. Τέλος η πύλη εξόδου, είναι αυτή που ελέγχει ποιες τιμές θα επιτραπούν ως έξοδοι από το μπλοκ.

Με βάση το σχήμα 25, παρατηρούμε το πώς λειτουργεί ένα μπλοκ ενός LSTM δικτύου, όπου η είσοδος x_t συνδέεται με την προηγούμενη έξοδο h_{t-1}. Στο πρώτο βήμα στο μπλοκ η συνδυασμένη τιμή εισόδου καλείται να σπάσει μέσω ενός στρώματος με συνάρτηση tanh.

$$g_t = \tanh\left(b^{(g)} + x_t U^{(g)} + h_{t-1} V^{(g)}\right)$$

Όπου U, V είναι τα βάρη της τιμής εισόδου και της προηγούμενης τιμής εξόδου αντίστοιχα, ενώ b η πόλωση της τιμής εισόδου.

Στο επόμενο βήμα, περνάει από την πύλη εισόδου, από όπου η τιμή εξόδου της πύλης πολλαπλασιάζεται με την τιμή που πέρασε από το πρώτο βήμα.

$$i_t = \sigma\left(b^{(i)} + x_t U^{(i)} + h_{t-1} V^{(i)}\right)$$

Η τιμή εξόδου στο βήμα αυτό δίνεται από

$$g_t \odot i_t,$$

Στη συνέχεια της ροής των δεδομένων στο μπλοκ, υπάρχει ο βρόγχος της πύλης forget. Στο μπλοκ υπάρχει η εσωτερική μεταβλητή s_t, αυτή η μεταβλητή με μία χρονική υστέρηση s_{t-1}, προστίθεται στα δεδομένα και δημιουργείται ένα στρώμα επανάληψης, το οποίο ελέγχεται από την πύλη forget.

Η πύλη forget εκφράζεται ως

$$f_t = \sigma\left(b^{(f)} + x_t U^{(f)} + h_{t-1} V^{(f)}\right)$$

Η έξοδος από την πύλη forget και το γινόμενο $g_t \odot i_t$, εκφράζεται ως $s_{t-1} \odot f_t$.

Ενώ η τελική τιμή εξόδου από τον βρόγχο επανάληψης της πύλης forget δίνεται από

$$s_t = s_{t-1} \odot f_t + g_t \odot i_t$$

Τέλος έχουμε το στρώμα εξόδου συνάρτησης tanh, το οποίο ελέγχεται από την πύλη εξόδου, η οποία εκφράζεται ως

$$o_t = \sigma\left(b^{(o)} + x_t U^{(o)} + h_{t-1} V^{(o)}\right)$$

Και η τελική τιμή εξόδου από το μπλοκ είναι

$$h_t = \tanh(s_t) \odot o_t$$

4.3.2 Παραλλαγές των LSTM Δικτύων.

Στην προηγούμενη παράγραφο, περιγράψαμε την διαδικασία ενός vanilla LSTM δικτύου, όμως υπάρχουν και κάποιες παραλλαγές τους οι οποίες χρησιμοποιούνται σε διάφορες εφαρμογές. Στη συνέχεια θα περιγράψουμε εν συντομία δύο βασικές από αυτές. (Μήτσιος, 2017)

LSTM with Peephole Connections

Αποτελεί μία από τις πιο δημοφιλής μεθόδους, όπου σύμφωνα με αυτή, προστίθενται στο μοντέλο συνδέσεις, οι οποίες αφήνουν τα στρώμματα των τριών πυλών να κοιτάξουν την κατάσταση του κελιού. Υπάρχουν περιπτώσεις που δεν χρησιμοποιούνται τέτοιες συνδέσεις και στις τρείς πύλες, αλλά σε μερικές από αυτές.

Ουσιαστικά στις εξισώσεις των πυλών που περιγράψαμε στην προηγούμενη παράγραφο, προστίθεται ό όρος για την peephole σύνδεση.

$$i = \sigma\left(b^{(i)} + x_t U^{(i)} + h_{t-1} V^{(i)} + p^{(i)} s_{t-1}\right)$$

$$f = \sigma\left(b^{(f)} + x_t U^{(f)} + h_{t-1} V^{(f)} + p^{(f)} s_{t-1}\right)$$

$$o = \sigma\left(b^{(o)} + x_t U^{(o)} + h_{t-1} V^{(o)} + p^{(o)} s_t\right)$$

Gated Recurrent Unit (GRU)

Η μέθοδος αυτή, παρουσιάστηκε από τον Kyunghyun Cho το 2014. Συνδυάζει τις πύλες εισόδου και forget σε μία, η οποία ονομάζεται πύλη ενημέρωσης *(update)*, ενώ η πύλη εξόδου, σε αυτή την παραλλαγή ονομάζεται πύλη επαναφοράς *(reset)*. Το μοντέλο που απορρέει από αυτή την τεχνική είναι αρκετά απλούστερο από το vanilla LSTM, αφού έχει λιγότερες παραμέτρους από αυτά.

Οι εξισώσεις που περιγράφουν τα GRU,

$$z_t = \sigma\left(b^{(z)} + x_t U^{(z)} + h_{t-1} V^{(z)}\right)$$

$$r_t = \sigma\left(b^{(r)} + x_t U^{(r)} + h_{t-1} V^{(r)}\right)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \sigma^{(h)}\left(b^{(h)} + x_t U^{(h)} + (r_t \odot h_{t-1}) V^{(h)}\right)$$

Όπου z_t και r_t, οι πύλες ενημέρωσης και επαναφοράς αντίστοιχα.

Οι τεχνικές που περιγράψαμε παραπάνω αποτελούν τις πιο διαδεδομένες έως τώρα, αλλά υπάρχουν περισσότερες. Για την ακρίβεια, μπορούν να γίνουν αρκετές μετατροπές στις πύλες του δικτύου και κάθε τέτοια μετατροπή δίνει μία διαφορετική εκδοχή των LSTM. Οι διαφορές τους είναι μικρές και δεν υπάρχει ένας γενικός κανόνας που καθιστά κάποιο από αυτά καλύτερο από τα άλλα. Σε γενικές γραμμές, το vanilla LSTM δείχνει πολύ καλή συμπεριφορά και απόδοση στις περισσότερες εφαρμογές και χρησιμοποιείται περισσότερο από τα υπόλοιπα.

4.3.3 Αλγόριθμος Βελτιστοποίησης LSTM – BackPropagation Through Time *(BPTT)*

Ο στόχος ενός αλγόριθμου BP, όπως αναφέραμε αναλυτικά σε προηγούμενη ενότητα, είναι να τροποποιήσει τα βάρη ενός νευρωνικού δικτύου, προκειμένου να ελαχιστοποιηθεί το σφάλμα των τιμών εξόδου του δικτύου συγκριτικά με κάποια αναμενόμενη τιμή εξόδου. Πρόκειται για έναν αλγόριθμο επόπτευόμενης μάθησης, να διορθωθεί σε σχέση με τα σφάλματα που προέκυψαν. (Brownlee, A Gentle Introduction to Backpropagation Through Time, 2017)

Ο αλγόριθμος BPTT, αποτελεί την μέθοδο βελτιστοποίησης για τα RNN που εφαρμόζονται κυρίως σε ακολουθιακά δεδομένα και είναι η συχνότερη μέθοδος συγκεκριμένα για τα LSTM. Σε ένα RNN, τα σφάλματα μπορούν να διαδοθούν περισσότερο, προκειμένου να καταγράψουν περισσότερες ιστορικές πληροφορίες. Η διαδικασία αυτή ονομάζεται ξεδίπλωση *(unfolding)*. Ο BPTT λειτουργεί ξεδιπλώνοντας όλες τις τιμές εισόδου σε κάθε χρονικό βήμα *(time step)*. Κάθε χρονικό βήμα έχει μία τιμή εισόδου, ένα αντίγραφο του δικτύου, και μία έξοδο. Στη συνέχεια υπολογίζονται και συσσωρεύονται σφάλματα για κάθε χρονική περίοδο. Το δίκτυο επαναφέρεται και τα βάρη ενημερώνονται.

Χωροταξικά, κάθε χρονομέτρηση του ξετυλιγμένου RNN, μπορεί να θεωρηθεί ως ένα πρόσθετο στρώμα δεδομένης της εξάρτησης του προβλήματος από την τάξη και η εσωτερική κατάσταση από το προηγούμενο χρονικό διάστημα λαμβάνεται ως είσοδος στο επόμενο χρονικό διάστημα.

Μπορούμε να συνοψίσουμε τα βήματα ως εξής:

1. Παρουσίαση μιας ακολουθίας χρονικών ζευγών εισόδου και εξόδου στο δίκτυο.
2. Ξεδίπλωση του δικτύου και στη συνέχεια υπολογισμός των σφαλμάτων σε κάθε χρονικό βήμα.
3. Ανάπτυξη του δικτύου και ενημέρωση των βαρών.
4. Επανέλαβε.

Ο αλγόριθμος αυτός όμως, παρά την ταχύτητα που προσφέρει, συγκριτικά με άλλες μεθόδους στην εκπαίδευση των RNN, φαίνεται να αντιμετωπίζει δυσκολίες κυρίως όσον αφορά την εύρεση του τοπικού βέλτιστου. Στα RNN, η εύρεση του τοπικού βέλτιστου αποτελεί σημαντικότερο πρόβλημα σε αντίθεση με τα feed-forward νευρωνικά δίκτυα. Επίσης άλλο ένα μειονέκτημα του, είναι ότι η μεταβολές στα βάρη, γίνονται στο τέλος κάθε epoch εκπαίδευσης, με αποτέλεσμα ιδιαίτερα σε εφαρμογές που τρέχουν σε πραγματικό χρόνο, να μην είναι ικανοποιητική η χρήση του.

Για τους λόγους αυτούς και όχι μόνο, έχουν εφευρεθεί και άλλες τεχνικές βελτιστοποίησης, οι οποίες χρησιμοποιούνται και στα LSTM, οι πιο βασικές εκ των οποίων είναι, η Extended Kalman Filtering *(EKF)* και η Real-Time Recurrent Learning *(RTRL).* (Jaeger, 2013)

Το RTRL, είναι ένας αλγόριθμος ο οποίος πραγματοποιεί την ενημέρωση των βαρών του δικτύου στο τέλος κάθε βήματος του αλγορίθμου κάτι το οποίο το καθιστά ιδιαίτερα χρονοβόρο, αλλά χρησιμοποιείται ευρέως σε εφαρμογές που τρέχουν σε πραγματικό χρόνο.

Το EKF αποτελεί μία γενίκευση του μοντέλου Kalman Filter για μη γραμμικά μοντέλα και επειδή χρησιμοποιεί μια γραμμική προσέγγιση ενός μη γραμμικού συστήματος, δεν προσφέρει καμία εγγύηση ότι θα καταφέρνει να βρίσκει σε όλες τις περιπτώσεις το ελάχιστο ενός σφάλματος. Παρόλα αυτά έχει αποδειχθεί χρήσιμη μέθοδος για την εκτίμηση των σφαλμάτων σε πολλές περιπτώσεις.

ΚΕΦΑΛΑΙΟ 5

Μέθοδοι Πρόβλεψης και Σχετικές Εργασίες

Τον ορισμό των τροχιών τον είχαμε ορίσει σε προηγούμενο κεφάλαιο, σε αυτό θα μελετήσουμε τις μεθόδους όσον αφορά την πρόβλεψη των τροχιών αρχικά, ενώ στη συνέχεια θα κάνουμε μία μικρή αναφορά σε ορισμένες εργασίες και θα μελετήσουμε τον τρόπο με τον οποίο προσπάθησαν να επιλύσουν το πρόβλημα της πρόβλεψης τροχιάς ενός κινούμενου αντικειμένου.

5.1 Μέθοδοι Πρόβλεψης

Οι μέθοδοι των μοντέλων για τα κινούμενα δεδομένα, χωρίζονται σε δύο βασικές κατηγορίες. (Georgiou, και συν., 2018)

Η πρώτη κατηγορία αφορά τα κινούμενα αντικείμενα όπου εντοπίζονται σε πραγματικό χρόνο για να παράγουν αναλυτικά στοιχεία και να υπολογίζουν βραχυπρόθεσμες προβλέψεις, οι οποίες είναι κρίσιμης σημασίας και απαιτούν άμεση ανταπόκριση. Η πρόβλεψη σε αυτές τις περιπτώσεις αφορά είτε το επόμενο σημείο του αντικειμένου, είτε ολόκληρη την τροχιά του, όπως θα δούμε και παρακάτω. Η βραχυπρόθεσμη πρόβλεψη θέσης και τροχιάς διευκολύνει τις αποτελεσματικές διαδικασίες σχεδιασμού, διαχείρισης και ελέγχου, ενώ αξιολογεί τις συνθήκες κίνησης π.χ. οδικές, θαλάσσιες και αεροπορικές μεταφορές. Το τελευταίο μπορεί να είναι εξαιρετικά σημαντικό σε τομείς όπου η ασφάλεια, η αξιοπιστία και το κόστος είναι κρίσιμοι παράγοντες.

Στη δεύτερη κατηγορία περιλαμβάνονται περιπτώσεις όπου οι μακροπρόθεσμες προβλέψεις είναι σημαντικές για τον εντοπισμό περιπτώσεων που δεν ακολουθούν τα κανονικά πρότυπα κίνησης, να ανιχνεύουν ανωμαλίες και να καθορίζουν μια θέση ή μια ακολουθία θέσεων σε ένα δεδομένο χρονικό διάστημα στο μέλλον. Σε αυτή την περίπτωση, παρόλο που ο χρόνος απόκρισης δεν είναι κρίσιμος παράγοντας, εξακολουθεί να είναι κρίσιμο για να προσδιοριστούν οι συσχετισμοί μεταξύ των ιστορικών προτύπων κινητικότητας και των προτύπων που αναμένεται να εμφανιστούν. Η μακροπρόθεσμη πρόβλεψη θέσης και τροχιάς μπορεί να ενισχύσει τα υπάρχοντα σχέδια για την επίτευξη οικονομικής απόδοσης ή ακόμα ανάλογα και των παρεχόμενων πληροφοριών (πχ καιρικές συνθήκες), να εξασφαλίσει τη δημόσια ασφάλεια μέσω διαφορετικών τρόπων μεταφοράς.

Παρακάτω παρατίθενται οι ορισμοί των δύο περισσότερο δημοφιλών προβλημάτων πρόβλεψης σε τροχιά. Το πρώτο είναι η πρόβλεψη του επόμενου σημείου ενός αντικειμένου, βασιζόμενο στην προηγούμενη θέση του και ονομάζεται Πρόβλεψη του Επόμενου Σημείου *(Future Location Prediction).* Ενώ το δεύτερο, αναφέρεται στην πρόβλεψη ολόκληρης της τροχιάς του αντικειμένου και ονομάζεται Πρόβλεψη Τροχιάς *(Trajectory Prediction).*

Ορισμός 1: Future Location Prediction
Δοθέντος μίας ανολοκλήρωτης τροχιάς $\{(p_0, t_0), (p_1, t_1), \dots, (p_{i-1}, t_{i-1})\}$ ενός κινούμενου αντικειμένου, αποτελούμενες από τις τοποθεσίες του κατά τις καταγεγραμμένες χρονικές σφραγίδες (timestamped locations) κατά τις i χρονικές στιγμές και μίας ακέραια τιμής j, προβλέπουμε την αναμενόμενη τροχιά του αντικειμένου, τις επόμενες j χρονικές στιγμές.

Ορισμός 2: Trajectory Prediction
Δοθέντος μιας ανολοκλήρωτης τροχιάς $\{(p_0, t_0), (p_1, t_1), \dots, (p_{i-1}, t_{i-1})\}$ ενός κινούμενου αντικειμένου, αποτελούμενες από τις τοποθεσίες του κατά τις καταγεγραμμένες χρονικές σφραγίδες (timestamped locations) κατά τις i χρονικές στιγμές και ενός σετ C από κάποιους περιορισμούς, προβλέπουμε την αναμενόμενη τροχιά του αντικειμένου, η οποία θα πρέπει να είναι συνεπείς με τους προαναφερθέντες περιορισμούς.

Επίσης, το πρώτο πρόβλημα, μπορεί να χωριστεί σε δύο επιμέρους κατηγορίες, όπου η πρώτη αναφέρεται σε μεθόδους οι οποίες βασίζονται σε τεχνικές μοτίβων *(pattern based)* και η δεύτερη αναφέρεται σε μεθόδους οι οποίες βασίζονται σε τεχνικές μοντέλων *(model based)*. Οι μέθοδοι βασισμένες σε μοτίβα εξάγουν μοτίβα κινητικότητας ενός αντικειμένου από ιστορικά δεδομένα και προβλέπουν την επόμενη τοποθεσία με βάση αυτά τα μοτίβα. Ενώ οι μέθοδοι που βασίζονται σε μοντέλα, εκπαιδεύουν στατιστικά μοντέλα, προκειμένου να παρατηρήσουν την τακτικότητα των κινήσεων και να κάνουν προβλέψεις βάση της εκμάθησης τους. (Yao, Zhang, Huang, & Bi, 2017)

Στην παρούσα εργασία και συγκεκριμένα στο επόμενο κεφάλαιο, θα προσπαθήσουμε να υλοποιήσουμε την μέθοδο του πρώτου ορισμού, με ορισμένες από τις τεχνικές που αναφέραμε σε προηγούμενα κεφάλαια.

Τα προαναφερθέντα και περισσότερες λεπτομέρειες σχετικά με αυτά, αναφέρονται στο paper των (Georgiou, και συν., 2018).

5.2 Σχετικές Εργασίες

Στο κομμάτι αυτό θα παρουσιάσουμε ορισμένες εργασίες οι οποίες προσπάθησαν να μελετήσουν το πρόβλημα της πρόβλεψης τροχιών με διάφορες μεθόδους.

Traj-ARIMA: A Spatial-Time Series Model for Network-Constrained Trajectory
Θα ξεκινήσουμε, παρουσιάζοντας το άρθρο του (Yan, 2010), στο οποίο προτείνει μία μέθοδο η οποία στοχεύει στην επέκταση του κλασικού μοντέλου ARIMA με χωρική διάσταση και την περαιτέρω εφαρμογή του για δεδομένα τροχιάς σε ένα περιορισμένο δίκτυο. Στην συγκεκριμένη εργασία προσπαθούν να επεκτείνουν τα κλασικά μοντέλα με σκοπό να τα χρησιμοποιήσουν για την πρόβλεψη της ταχύτητας αντικειμένων σε ένα περιορισμένο δίκτυο.

Όπως έδειξαν οι Box and Jenkins, οι χρονοσειρές αποτελούν στοχαστικές διαδικασίες από στατιστικής άποψης. Τα ARIMA αποτελούν την σημαντικότερη γραμμική μέθοδο για την εκτίμηση χρονοσειρών. Ένα μοντέλο ARIMA, ουσιαστικά αποτελεί συνδυασμό ενός αυτοπαλίνδρομου *(AR)* μοντέλου και ενός μοντέλου κινητού μέσου *(MA)*. Το αυτοπαλίνδρομο μοντέλο, αποτελεί ουσιαστικά μία παλινδρόμηση, πράγμα που σημαίνει ότι η τρέχουσα τιμή x_t είναι ένας γραμμικός συνδυασμός από p ιστορικές παρατηρήσεις και τον λευκό θόρυβο e_t. Το μοντέλο κινούμενου μέσου τάξης q αποτελεί γραμμικό συνδυασμό του τρέχοντος θορύβου και των q ιστορικών θορύβων. Το μοντέλο ARMA, συνδυάζει τα προηγούμενα μοντέλα.

$$AR(p): x_t = \varphi_1 x_{t-1} + \cdots + \varphi_p x_{t-p} + e_t = \sum_{i=1}^{p} \varphi_i x_{t-i} + e_t$$

$$MA(q): x_t = e_t + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} = e_t + \sum_{j=1}^{q} \theta_j e_{t-j}$$

$$ARMA(p,q): x_t = \varphi_1 x_{t-1} + \cdots + \varphi_p x_{t-p} + e_t + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q}$$
$$= \sum_{i=1}^{p} \varphi_i x_{t-i} + e_t + \sum_{j=1}^{q} \theta_j e_{t-j}$$

Οι παραπάνω εξισώσεις μπορούν να γραφούν και στη μορφή χρησιμοποιώντας backshift operator B

$$AR(p): x_t\left(1 - \varphi_1 B - \varphi_2 B^2 - \cdots - \varphi_p B^p\right) = x_t\left(1 - \sum_{i=1}^{p} \varphi_i B^i\right) = e_t$$

$$MA(q): x_t = e_t(1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q) = e_t\left(1 + \sum_{j=1}^{q} \theta_j B^j\right)$$

$$ARMA(p,q): x_t\left(1 - \sum_{i=1}^{p} \varphi_i B^i\right) = e_t\left(1 + \sum_{j=1}^{q} \theta_j B^j\right)$$

Τα μοντέλα ARMA μπορούν να επεκταθούν σε ARIMA, παίρνοντας πρώτες διαφορές, προκειμένου να μετατραπεί το μοντέλο σε στάσιμο αν δεν είναι.

$$ARIMA(p,d,q): x_t\left(1 - \sum_{i=1}^{p} \varphi_i B^i\right)(1-B)^d = e_t\left(1 + \sum_{j=1}^{q} \theta_j B^j\right)$$

Το μοντέλο αυτό, λαμβάνει υπόψη του μονάχα τις χρονικές συσχετίσεις των παρατηρήσεων της χρονοσειράς, για δεδομένα τροχιάς όμως, είναι απαραίτητο να λάβουμε υπόψη και τις χωρικές συσχετίσεις. Υπάρχουν δύο μέθοδοι που προεκτείνουν την προηγούμενη και λαμβάνουν υπόψη την χωρική διάσταση, τα Vector ARMA και τα Space-Time ARIMA.

Τα Vector ARMA εκτιμούν τις δυναμικές αλληλεπιδράσεις μεταξύ πολλαπλών χρονολογικών σειρών, οι οποίες μπορούν να ληφθούν υπόψη ως υποκατηγορία των χώρο-χρονικών μοντέλων. Η διαφορά με τα κλασικά ARMA μοντέλα, έγκειται στο γεγονός ότι αλλάζουν το διάνυσμα σε πολυδιάστατο.

$$x_t = \Phi_1 X_{t-1} + \cdots + \Phi_p X_{t-p} + e_t + e_t + \Theta_1 e_{t-1} + \cdots + \Theta_q e_{t-q} = \sum_{i=1}^{p} \Phi_i X_{t-i} + e_t + \sum_{j=1}^{q} \Theta_j e_{t-j}$$

Όπου, $X_t = (x_{t1}, \dots, x_{tk})'$, $e = (e_{t1}, \dots, e_{tk})'$, είναι διανύσματα των παρατηρήσεων και λευκός θόρυβος αντίστοιχα, ενώ Φ_p και Θ_q είναι πίνακες συντελεστών που πρέπει να εκτιμηθούν.

Από την άλλη μεριά τα Space-Time ARIMA, μπορούν να θεωρηθούν μία ειδική περίπτωση των vector ARMA, δίνοντας έμφαση στην χωρική συσχέτιση των παρατηρήσεων πέρα της χρονικής. Για την

ακρίβεια, εκφράζουν κάθε παρατήρηση στον χρόνο t και στο χώρο l ως γραμμικό συνδυασμό των προηγούμενων παρατηρήσεων με τόσο χρονικές όσο και χωρικές υστερήσεις. Δοθέντος μίας παρατήρησης $X_t = (x_{t1}, \dots, x_{tl})'$ με l παρατηρούμενες τιμές στον χρόνο t σε l διαφορετικές τοποθεσίες και $W = \left[w_{ij}\right]_{l*l}$ τον πίνακα $l * l$ διαστάσεων, για τις l σχετικές τοποθεσίες, εκφράζεται το παρακάτω ST-ARMA μοντέλο.

$$X_t = \sum_{j=1}^{p}\sum_{k=1}^{l} \Phi_i W_k X_{t-i} + e_t + \sum_{j=1}^{q}\sum_{k=1}^{l} \Theta_j W_k e_{t-j}$$

Το οποίο μπορεί να γραφεί

$$\left(I - \sum_{j=1}^{p}\sum_{k=1}^{l} \Phi_i W_k B^i\right) X_t = e_t \left(I + \sum_{j=1}^{q}\sum_{k=1}^{l} \Theta_j W_k B^j\right)$$

Και μπορεί να επεκταθεί σε ένα ST-ARIMA μοντέλο

$$\left(I - \sum_{j=1}^{p}\sum_{k=1}^{l} \Phi_i W_k B^i\right)\left(1 - B^d\right) X_t = e_t \left(I + \sum_{j=1}^{q}\sum_{k=1}^{l} \Theta_j W_k B^j\right)$$

Προχωρώντας στην υλοποίηση του μοντέλου, ο συγγραφέας ξεκινάει μετατρέποντας τα δεδομένα τροχιών σε ένα συμπαγές μοντέλο χρονολογικών σειρών. Υπολογίζουν την στιγμιαία ταχύτητα του αντικειμένου ως μέσο όρο μεταξύ της προηγούμενης και της επόμενης χρονικής στιγμής από τα δεδομένα.

$$s_i = \frac{\|(x_{i+1}, y_{i+1}) - (x_{i-1}, y_{i-1})\|_2^2}{t_{i+1} - t_{i-1}}$$

Και με αυτόν τον τρόπο λαμβάνουν την χρονολογική σειρά ταχύτητας του κάθε αντικειμένου $\left(s_{i,} t_i\right)$. Έτσι μπορεί να χτιστεί το μοντέλο ARMA για την χρονολογική σειρά αυτή.

$$s_t \left(1 - \sum_{i=1}^{p} \varphi_i B^i\right) = e_t \left(1 + \sum_{j=1}^{q} \theta_j B^j\right)$$

Στη συνέχεια προκειμένου να επεκταθούν στο ST-ARMA μοντέλο, πρέπει να λάβουν υπόψη τις χωρικές συσχετίσεις, οι οποίες όμως είναι δυναμικές και επηρεάζονται από το δίκτυο, έτσι πρέπει να κατασκευάσουν μία ακόμα χρονολογική σειρά για την ταχύτητα σε γειτονικούς δρόμους και την ονομάζουν *ροή τροχιάς.*

$$s_t = F(s_{t-1}, s_{t-2}, \dots \quad , \quad f_{rk}, f_{rk-1}, \dots.)$$

Όπου $s_{t-1}, s_{t-2}, \dots$ είναι οι ιστορικές ταχύτητες με χρονικές συσχετίσεις και $f_{rk}, f_{rk-1}, \dots.$ είναι οι ροές τροχιάς των γειτονικών δρόμων με χωρικές συσχετίσεις.

Και με αυτόν τον τρόπο λαμβάνουν την χρονολογική σειρά της ροής τροχιάς $F = \{(t_i, f_i)\}$. Έτσι μπορεί να χτιστεί το μοντέλο βασισμένο στο ST-ARIMA για τη χρονολογική σειρά αυτή.

$$\left(I - \sum_{j=1}^{p}\sum_{k=1}^{l} \Phi_i W_k B^i\right)\left(1 - B^d\right) F_t = e_t \left(I + \sum_{j=1}^{q}\sum_{k=1}^{l} \Theta_j W_k B^j\right)$$

Έτσι, χρειάζεται να χρησιμοποιήσουν την χρονολογική σειρά της ροής τροχιάς για να μοντελοποιήσουν την χρονολογική σειρά της ταχύτητας. Για να γίνει αυτό, πρέπει να συνδυάσουν τις δύο προηγούμενες εξισώσεις, κάτι το οποίο μπορεί να γίνει με τρείς τρόπους.

1. Διαδικασία χρονολογικής σειράς ταχύτητας τροχιάς και ταχύτητας ροής τροχιάς μαζί.

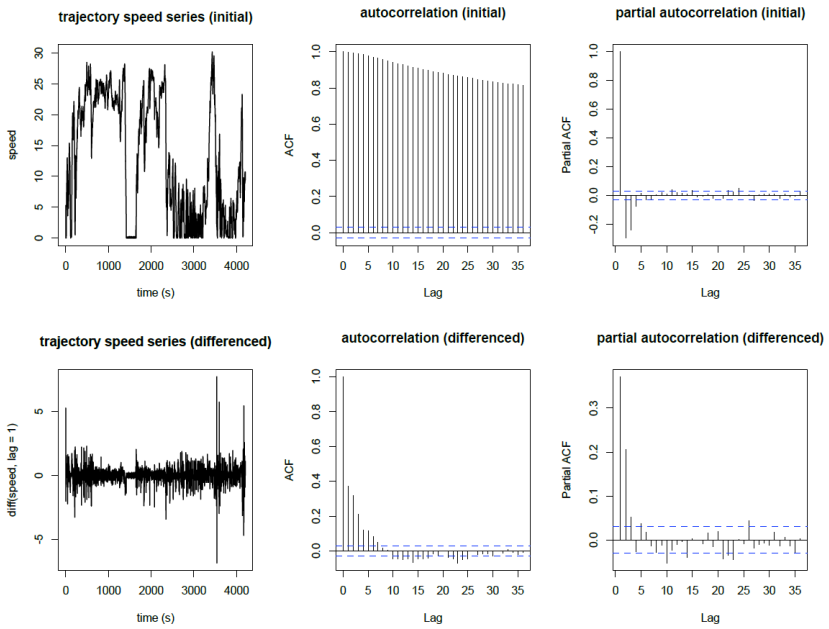
$$\left(I - \sum_{j=1}^{p}\sum_{k=1}^{l} \Phi_i W_k B^i\right)(1 - B^d)X_t = e_t\left(I + \sum_{j=1}^{q}\sum_{k=1}^{l} \Theta_j W_k B^j\right)$$

2. Ξεχωριστά να χτίζει εκ των προτέρων τις χρονολογικές σειρές ροής τροχιάς και στη συνέχεια να τις συνδυάζει γραμμικά στο μοντέλο της σειράς ταχύτητας της τροχιάς.

$$\left(1 - \sum_{i=1}^{p} \varphi_i B^i\right)(1 - B)^d\left(s_t + \sum Wf\right) = e_t\left(1 + \sum_{j=1}^{q} \theta_j B^j\right)$$

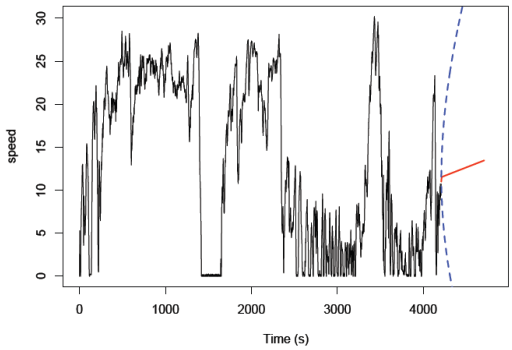
3. Περαιτέρω προέκταση του προηγούμενου με εξέταση των δυναμικά χωρικών υστερήσεων για την τροχιά, σαν να συμμετέχουν διαφορετικά κομμάτια δρόμου με την εξέλιξη της τροχιάς.

Όσον αφορά την υλοποίηση, χρησιμοποιήθηκε πραγματικό σετ δεδομένων από τροχιές αυτοκινήτων στην Βραζιλία. Από τον πίνακα αυτοσυσχετίσεων, παρατηρείται ότι χρειάστηκε να πάρουν πρώτες διαφορές, προκειμένου να μετατρέψουν την χρονολογική σειρά σε στάσιμη. Τα αποτελέσματα φαίνονται στα παρακάτω γραφήματα.



Σχήμα 26

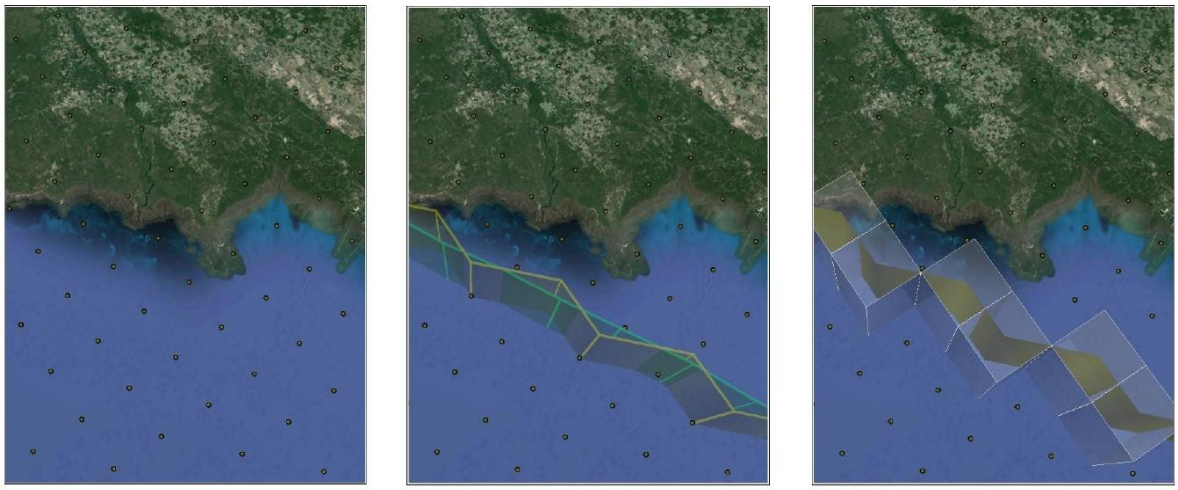
Από τα γραφήματα αυτά προέκυψε επίσης ότι το καταλληλότερο μοντέλο είναι ένα ARIMA (2,1,6). Χρησιμοποιώντας αυτό το μοντέλο, προέκυψε η πρόβλεψη, όπως παρουσιάζεται στο παρακάτω γράφημα.



Σχήμα 27: Πρόβλεψη Μοντέλου

Aircraft Trajectory Prediction Made Easy with Predictive Analytics

Οι (Ayhan & Samet, 2016) στην εργασία τους Aircraft Trajectory Prediction Made Easy with Predictive Analytics, περιγράφουν μία διαφορετική τεχνική για την πρόβλεψη τροχιάς αεροσκαφών, η οποία σκοπεύει σε πιο ρεαλιστικές και ακριβής μεθόδους σχεδιασμού πτήσεων, ώστε να βελτιωθεί η ασφάλεια και η οικονομία στις πτήσεις. Στην προσέγγιση τους, σχεδιάζουν τον εναέριο χώρο ως ένα 3D δίκτυο πλέγματος, όπου κάθε σημείο πλέγματος αποτελεί την τοποθεσία μίας καιρικής παρατήρησης. Στη συνέχεια, χτίζουν υποθετικά κύβους γύρω από αυτά τα σημεία του πλέγματος, θεωρώντας όλο τον εναέριο χώρο ως ένα σύνολο κύβων. Οι καιρικές συνθήκες παραμένουν σταθερές σε κάθε κύβο που δημιουργείται. Έπειτα, συνδυάζοντας τους κύβους με τις τροχιές, δημιουργούν ένα 4D πλέγμα. Σε αυτό το πλέγμα εκπαιδεύουν το μοντέλο τους βάσει των ιστορικών γεγονότων, σε συνάρτηση των καιρικών συνθηκών και χτίζουν το μοντέλο τους για να βρεθεί η πιθανή τροχιά του αεροσκάφους σε κάθε κύβο.



Σχήμα 28: Αναπαράσταση του πλέγματος στον εναέριο χώρο
(Ayhan & Samet, 2016)

Το μοντέλο το οποίο χρησιμοποιούν είναι ένα Hidden Markov Model. Δεδομένης μιας σειράς παρατηρήσεων, προσπαθούν να εκπαιδεύσουν ένα HMM, ένα στατιστικό μοντέλο Markov, και να αντλήσουν μία ακολουθία κρυφών καταστάσεων που αντιστοιχούν στην ακολουθία. Για να το υπολογίσουν αυτό, χρησιμοποιούν έναν αλγόριθμο Viterbi.

Ο αλγόριθμος Viterbi, είναι ένας δυναμικός προγραμματιστικός αλγόριθμος, σχεδιασμένος για να βρίσκει την πιο πιθανή ακολουθία από κρυμμένες καταστάσεις *(Viterbi Path)*, που έχουν ως αποτέλεσμα παρατηρούμενα γεγονότα. (wikipedia, n.d.)

Ο αλγόριθμος δημιουργεί ένα μονοπάτι $X = (x_1, \dots, x_T)$, το οποίο αποτελεί μία ακολουθία από καταστάσεις $x_n \in S = (s_1, \dots, s_K)$, παραγόμενες από τις παρατηρήσεις $Y = (y_1, \dots, y_T)$, με $y_n \in O = (o_1, \dots, o_N)$. Ας υποθέσουμε ότι έχουμε ένα HMM με αρχικές πιθανότητες π_i να βρίσκεται στην κατάσταση i και πιθανότητα μετάβασης από την κατάστασή i στην j , $a_{i,j}$. Αν παρατηρούμε τιμές εξόδου $y_1, \dots, y_T$, τότε η πιθανότερη ακολουθία $x_1, \dots, x_T$ από την οποία προέκυψαν οι παρατηρήσεις, δίνεται από τις παρακάτω σχέσεις:

$$V_{1,k} = P(y_1|k)\pi_k$$

$$V_{t,k} = max_{x \in S}\left(P(y_t|k)a_{x,k}V_{t-1,x}\right)$$

Όπου $V_{t,k}$ είναι η πιθανότητα της πιο πιθανής ακολουθίας $P(x_1, \dots, x_t, y_1, \dots, y_t)$ για τις πρώτες t παρατηρήσεις, που έχουν k ως τελική κατάστασή τους. Το μονοπάτι Viterbi υπολογίζεται με ένα βήμα προς τα πίσω, το οποίο θυμάται ποια κατάσταση x χρησιμοποιήθηκε στην δεύτερη εξίσωση.

$$x_t = \arg max_{x \in S}(V_{t,x})$$

$$x_{t-1} = Ptr(x_t, t)$$

Όπου Ptr η συνάρτηση η οποία επιστρέφει την τιμή x που χρησιμοποιήθηκε στον υπολογισμό του $V_{t,k}$.

Συνοψίζοντας, η εργασία τους διαφοροποιείται σε σχέση με προηγούμενες εργασίες στα εξής:

- Χρησιμοποιούν μία πιθανολογική προσέγγιση, λαμβάνοντας υπόψη διάφορες αβεβαιότητες, έτσι επιτυγχάνουν μεγαλύτερη ακρίβεια στο μοντέλο τους.
- Θεωρούν τις τροχιές σαν ένα σετ από 4D ενωμένους κύβους, το οποίο τους βοηθάει να χωρίσουν τις τροχιές σε τμήματα ανάλογα με τα καιρικά φαινόμενα.
- Πραγματοποιούν τεχνική συσταδοποίησης χρονοσειρών της οποίας τα αποτελέσματα τροφοδοτούνται στον αλγόριθμο Viterbi.
- Χρησιμοποιούν πραγματικά δεδομένα και καιρικές συνθήκες για να επιτύχουν μεγαλύτερη αποτελεσματικότητα.

Στη συνέχεια θα παρουσιάσουμε κάποιες εργασίες, οι οποίες βασίστηκαν σε νευρωνικά δίκτυα για την πρόβλεψη τροχιάς.

SERM: A Recurrent Model for Next Location Prediction in Semantic Trajectories

Αρχικά θα παρουσιάσουμε την εργασία των (Yao, Zhang, Huang, & Bi, 2017), οι οποίοι στο paper τους παρουσίασαν την μέθοδο SERM *(Semantic Enriched Recurrent Model)* για την πρόβλεψη της επόμενης θέσης ενός κινούμενου αντικειμένου. Οι συγγραφείς του συγκεκριμένου paper, συνοψίζουν τις διαδικασίες που ακολούθησαν ως εξής:

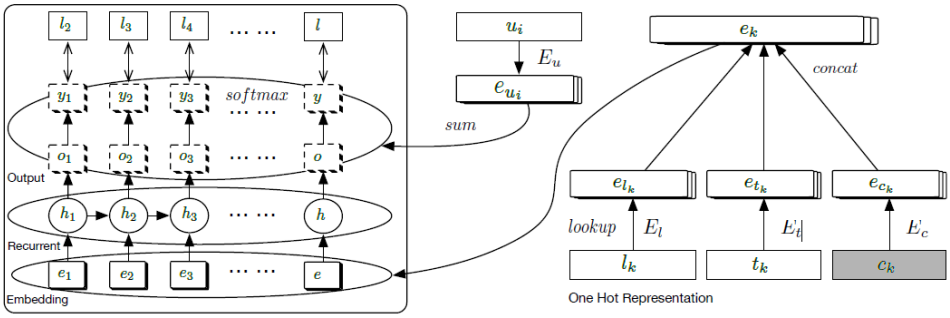
- Δοκιμάζουν πέρα από την απλή πρόβλεψη της επόμενης θέσης για τροχιές προερχόμενες από GPS, μελετάνε την πρόβλεψη για εμπλουτισμένες *(semantic)* τροχιές.
- Προτείνουν, ένα εμπλουτισμένο οπισθοδρομικό *(recurrent)* μοντέλο, το οποίο εκπαιδεύεται από κοινού με διάφορους παράγοντες και παραμέτρους μετάβασης ενός οπισθοδρομικού δικτύου.

- Πειραματίζονται σε δύο πραγματικά σετ δεδομένων και παρατηρούν ότι η μέθοδο τους υπερτερεί από προγενέστερες μεθόδους, όπως θα δούμε παρακάτω.

Στην αρχή παρουσιάζουν τους ορισμούς για το τι είναι τροχιά και τι είναι σημασιολογική τροχιά, στους οποίους έχουμε αναφερθεί σε προηγούμενες ενότητες. Έπειτα κάνουν τις εξής παρατηρήσεις:

- Η κίνηση ενός αντικειμένου, υπόκειται στις χώρο-χρονικές κανονικότητες *(νόμους)*.
- Δεύτερον, η σημασιολογία της τρέχουσας δραστηριότητας ενός αντικειμένου χρησιμεύει ως ένα χρήσιμο σήμα για την επόμενη πρόβλεψη θέσης.
- Τέλος, ο χρήστης οι προσωπικές προτιμήσεις παίζουν συχνά σημαντικό ρόλο στην πρόβλεψη της επόμενης θέσης.

Για να εφαρμόσουν τις παραπάνω παρατηρήσεις τους, παρουσιάζουν το μοντέλο, όπως φαίνεται στην εικόνα παρακάτω, όπου ενσωματώνουν τους διαφορετικούς παράγοντες σε ένα πολυάριθμο προς πολυάριθμο *(many to many)* οπισθοδρομικό νευρωνικό δίκτυο.



Σχήμα 29: Αναπαράσταση Μοντέλου

Για περισσότερες λεπτομέρειες σχετικά με το χτίσιμο των στρωμάτων, καλείται ο αναγνώστης να αναφερθεί στο συγκεκριμένο κείμενο των συγγραφέων.

Λαμβάνοντας υπόψιν την τροχιά ενός χρήστη $T_u(u_i) = \{r_1(u_i), \dots, r_k(u_i)\}$, το στρώμα ενσωμάτωσης *(embedding layer)*, είναι σχεδιασμένο να καταγράφει την πληροφορία από τους δυναμικούς παράγοντες σε κάθε καταγραφή $r_k(u_i) = (t_k, l_k, c_k)$.

Για κάθε καταγραφή $r_k(u_i)$ στην τροχιά εισόδου, οι μονάδες ενσωμάτωσης είναι ικανές να παράξουν αντιπροσωπευτικά διανύσματα $e_k \in R^{De}$ για κάθε εγγραφή (t_k, l_k, c_k). Για μία k μήκους σημασιολογική τροχιά, το στρώμα οπισθοδρόμησης, αποτελείται από k επαναλαμβανόμενες *(recurrent)* μονάδες. Η $k-$οστή μονάδα ενσωματώνει την καταγραφή των (t_k, l_k, c_k) και υπολογίζει την κρυμμένη κατάσταση h_k.

Η κρυμμένη κατάσταση h_k, κωδικοποιεί την παρατηρούμενη πληροφορία μέχρι το βήμα k. Έπειτα ενσωματώνει τις προσωπικές προτιμήσεις των χρηστών για την εξαγωγή της πρόβλεψης.

Οι συγγραφείς πειραματίστηκαν με δύο σετ δεδομένων και σε αυτά μαζί με τη μέθοδο που παρουσιάζουν στο paper τους, κάνουν σύγκριση με τις μεθόδους Nearest Location, Hidden Markov Models, ST-RNN. Τα αποτελέσματα, φαίνονται στον παρακάτω πίνακα.

Data	Method	HR@1	HR@5	HR@10	HR@20	δ_d/m
NY	NL	0.1630	0.2455	0.2998	0.4386	2903
	MF	0.1690	0.4326	0.5013	0.5358	1963
	HMM	0.1763	0.4298	0.5251	0.5518	1952
	ST-RNN	0.1942	0.4421	0.5381	0.6053	1602
	SERM*	0.2181	0.4398	0.5401	0.6107	1563
	SERM	**0.2535**	**0.4507**	**0.5433**	**0.6237**	**1457**
LA	NL	0.3745	0.4516	0.4704	0.4911	6061
	MF	0.3646	0.5810	0.6354	0.6877	2647
	HMM	0.3921	0.5935	0.6331	0.6732	2521
	ST-RNN	0.4311	0.6013	0.6521	0.6980	2384
	SERM*	0.4452	0.6147	0.6590	0.6973	2377
	SERM	**0.4625**	**0.6265**	**0.6670**	**0.7026**	**2177**

Σχήμα 30: Σύγκριση απόδοσης μεθόδων. HR είναι το hitting ratio και δ_d είναι το σφάλμα πρόβλεψης στην απόσταση

Σε αυτόν τον πίνακα παρατηρούμε ότι η μέθοδο SERM που προτείνουν οι συγγραφείς του paper, δίνουν καλύτερα αποτελέσματα σε σύγκριση με τις άλλες μεθόδους που χρησιμοποίησαν.

A Seq2seq Learning Approach for Modeling Semantic Trajectories and Predicting the Next Location.

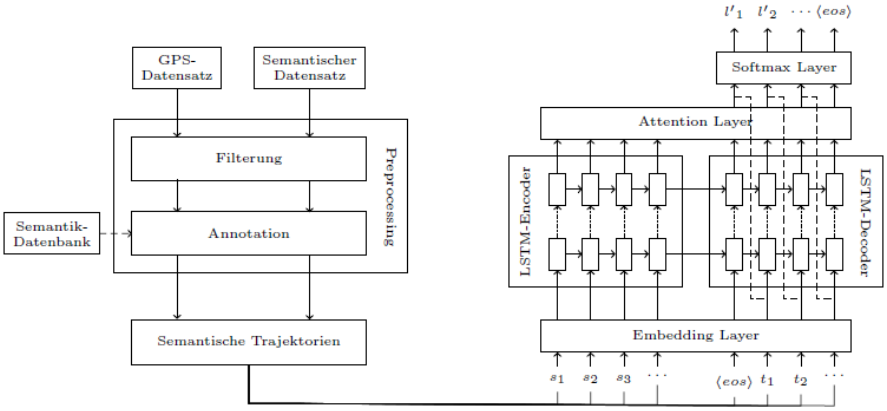
Τέλος, θα αναφερθούμε στο paper των (Karatzoglou, Jablonski, & Beigl, 2018), όπου προτείνουν την μέθοδο εκμάθησης Sequence to Sequence, η οποία είναι ευρέως διαδεδομένη στην εξόρυξη γνώσης κειμένων. Οι συγγραφείς σε αυτό το paper, χρησιμοποιούν την μέθοδο αυτή για την πρόβλεψη της μελλοντικής κίνησης χρηστών. Η μέθοδος αυτή είναι σχεδιασμένη να εκπαιδεύεται έτσι ώστε να παράγει ακολουθίες εξόδου, βάσει των ακολουθιών εισόδου που δέχεται.

Η αρχιτεκτονική της μεθόδου αυτής, αποτελείται από δύο τμήματα. Τον κωδικοποιητή *(encoder)* και τον αποκωδικοποιητή *(decoder),* οι οποίοι μπορεί να είναι είτε ένα απλό RNN, είτε ένα LSTM. Ο κωδικοποιητής, είναι αυτός που δέχεται την ακολουθία εισόδου και παράγει αναπαράσταση της ως έξοδο στο τελευταίο στρώμα h_n και στο κελί μνήμης c_n αν πρόκειται για LSTM δίκτυο. Ο αποκωδικοποιητής με τη σειρά του, λαμβάνει την έξοδο του κωδικοποιητή και παράγει βήμα βήμα την ακολουθία που θέλουμε να προβλέψουμε.

Οι μακριές ακολουθίες εισόδου μπορούν να οδηγήσουν το μοντέλο Seq2Seq σε κώλυμα όσον αφορά την απόδοση της μοντελοποίησης. Για να το αποφύγουν αυτό, οι συγγραφείς του συγκεκριμένου paper, προτείνουν τον μηχανισμό προσοχής *(attention mechanism).* Η μέθοδος αυτή επεκτείνει εκείνη των Seq2Seq μοντέλων, δίνοντας την κατάλληλη πληροφορία σε αυτό, για το που θα πρέπει να δοθεί προσοχή. Συγκεκριμένα, ο μηχανισμός προσοχής ενημερώνει τον αποκωδικοποιητή για τις θέσεις των στοιχείων, όπου η πιο σημαντική πληροφορία θα μπορούσε να εντοπιστεί σε κάθε βήμα του χρόνου. Με αυτόν τον μηχανισμό, βελτιώνεται η απόδοση του μοντέλου καθώς και η ακρίβεια της πρόβλεψης.

Το πρώτο αφορά το κομμάτι της προ-επεξεργασίας και το δεύτερο της πρόβλεψης. Όσον αφορά το κομμάτι της προ-επεξεργασίας, καθαρίζουν όλες τις διπλές και ανακριβείς καταγραφές. Στη συνέχεια χρησιμοποιούν τον αλγόριθμο DBSCAN, για να ομαδοποιήσουν τα δεδομένα, ενώ στο τέλος όλες οι σημασιολογικές ακολουθίες που έχουν εξαχθεί προωθούνται για το κομμάτι της εκπαίδευσης. Για το σκοπό αυτό, κάθε σημασιολογική θέση λαμβάνει ένα μοναδικό αναγνωριστικό *(ID)* και μορφοποιείται ως κωδικοποιημένο διάνυσμα.

Το κομμάτι της εκπαίδευσης και πρόβλεψης, πρόκειται να εκπαιδεύσει ένα Seq2Seq μοντέλο βασισμένο στις παρεχόμενες σημασιολογικές τροχιές και στη συνέχεια για την δημιουργία προβλέψεων σχετικά με τις μελλοντικές τοποθεσίες των χρηστών.

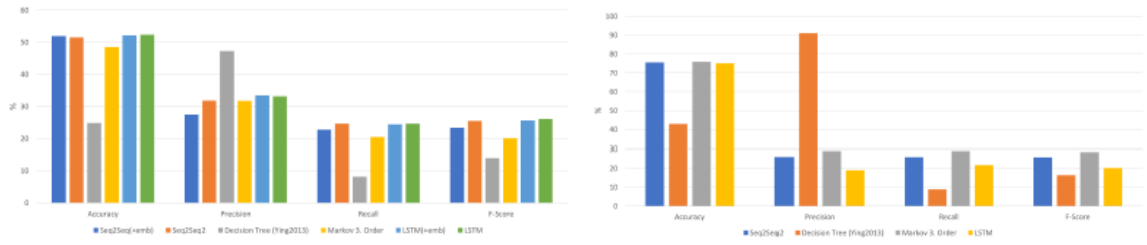


Σχήμα 31: Αναπαράσταση Μοντέλου

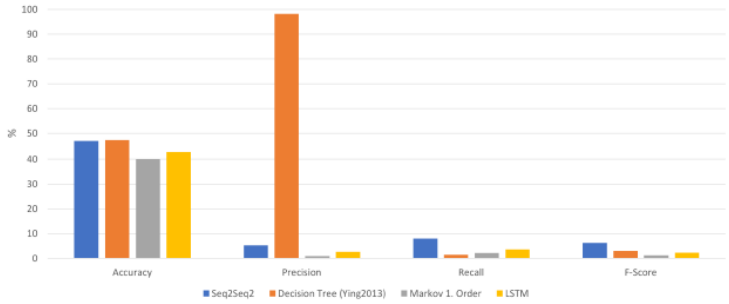
Το μοντέλο αποτελείται από τα ακόλουθα στοιχεία: το στρώμα ενσωμάτωσης *(embedding)*, κωδικοποιητή *(encoder)*, αποκωδικοποιητή *(decoder)*, το στρώμα προσοχής *(attention layer)* και το στρώμα εξόδου αποτελούμενο από τη συνάρτηση Softmax. Στόχος του στρώματος ενσωμάτωσης, είναι να παράξει πιο πυκνές αντιπροσωπεύσεις των διανυσμάτων, ώστε να βελτιωθεί η απόδοση του μοντέλου. Ο κωδικοποιητής αποτελείται από ένα πολύ-στρωματικό LSTM δίκτυο, το οποίο διαβάζει την ακολουθία εισόδου βήμα προς βήμα, ενώ ο αποκωδικοποιητής, αποτελείται επίσης από ένα πολύ-στρωματικό LSTM δίκτυο, με κύριο καθήκον του να παράξει την ακολουθία εξόδου. Το στρώμα προσοχής *(attention layer)*, παίρνει τα διανύσματα εξόδου από τα προηγούμενα στρώματα και χτίζει ένα σταθερού μήκους διάνυσμα α. Τέλος, το διάνυσμα αυτό περνάει στο στρώμα εξόδου, το οποίο υπολογίζει τις επόμενες ακολουθίες, από τις οποίες ενδιαφερόμαστε για το πρώτο στοιχείο τους, που αποτελεί την επόμενη θέση του αντικειμένου $\hat{l}_i$. Αυτό υπολογίζεται βάσει μίας συνάρτησης Softmax.

$$\hat{l}_i = softmax(W_s * a_i)$$

Το μοντέλο που δημιουργήθηκε, συγκρίθηκε με ένα απλό μοντέλο LSTM δικτύων, ένα μοντέλο Hidden Markov, όπως επίσης και με ένα μοντέλο βασισμένο σε δένδρα απόφασης. Τα αποτελέσματα φαίνονται στα παρακάτω γραφήματα. Γίνεται εφαρμογή σε δύο διαφορετικά σετ δεδομένων.



Σχήμα 32: Αριστερά, απόδοση μοντέλων στην περίπτωση της πρόβλεψης πολλαπλών χρηστών, δεξιά, απόδοση μοντέλων στην περίπτωση της πρόβλεψης απλού χρήστη, στο MIT σετ δεδομένων.



Σχήμα 33: Απόδοση μοντέλων στην περίπτωση της πρόβλεψης πολλαπλών χρηστών στο SDP σετ δεδομένων.

Συνοψίζοντας, παρατηρώντας και τα παραπάνω γραφήματα, παρατηρούμε ότι η μέθοδος αυτή παρήγαγε αρκετά καλά αποτελέσματα, αλλά δεν παρουσίασε μεγάλη διαφορά στα αποτελέσματα σε σύγκριση με τις άλλες μεθόδους που χρησιμοποιήθηκαν. Επίσης, όπως παρατηρούμε, δοκίμασαν και τη χρήση ενός αυτό-δίδακτου στρώματος ενσωμάτωσης στα LSTM και στα Seq2Seq LSTM, το οποίο συνήθως βοηθάει την βελτίωση της απόδοσης των δικτύων. Βέβαια όπως φαίνεται στα παραπάνω γραφήματα, στην περίπτωση αυτή δεν βελτιώνει ιδιαίτερα τα αποτελέσματα.

ΚΕΦΑΛΑΙΟ 6

Υλοποίηση

6.1 Σκοπός

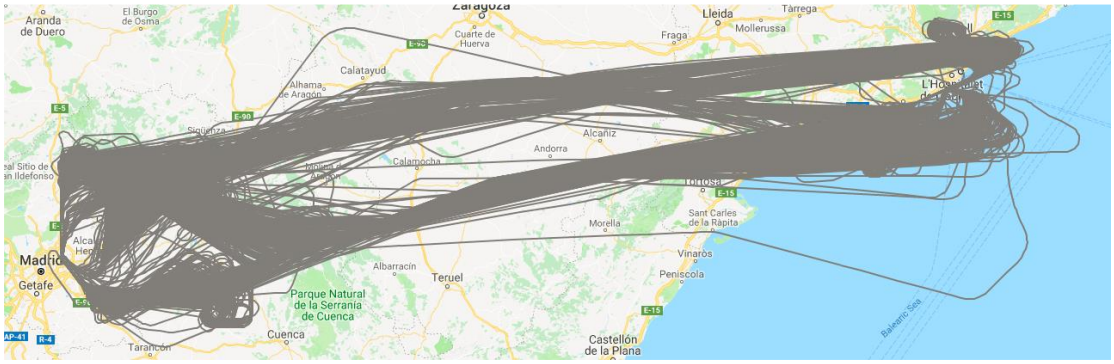
Στο κεφάλαιο αυτό θα περιγράψουμε και θα παρουσιάσουμε την ανάλυση που εφαρμόσαμε, με σκοπό την πρόβλεψη τροχιάς ενός αεροπλάνου. Για την ακρίβεια εφαρμόσαμε μία βραχυπρόθεσμη μέθοδο πρόβλεψης, στην οποία βασιστήκαμε στην τωρινή θέση του αεροπλάνου, προκειμένου να προβλέψουμε την θέση την οποία αυτό θα έχει την επόμενη χρονική στιγμή.

Η εφαρμογή του προβλήματος αυτού, έγινε μέσω της γλώσσας προγραμματισμού Python και συγκεκριμένα εφαρμόσαμε μεθόδους νευρωνικών δικτύων, μέσω της βιβλιοθήκης Keras. Η βιβλιοθήκη Keras, είναι μία πού εύχρηστη βιβλιοθήκη, η οποία δημιουργήθηκε μέσω της έρευνας για τον έργο ONEIROS *(Open-ended Neuro-Electronic Intelligent Robot Operating System*) και πήρε την ονομασία της από την ελληνική λέξη κέρατο.

Όσον αφορά τα δεδομένα τα οποία χρησιμοποιήσαμε, αυτά εμπεριέχουν όλες τις πτήσεις που έγιναν μεταξύ Μαδρίτης και Βαρκελώνης κατά τη διάρκεια του μηνός Απριλίου του 2016. Τα δεδομένα προέρχονται από τα ραντάρ IFS τα οποία παρέχει η CRIDA και συγκεκριμένα περιλαμβάνουν 909.644 θέσεις πρωταρχικών σημείων από 1396 πτήσεις αεροσκαφών. Τα δεδομένα μας, είναι ακολουθιακά και περιέχουν τις κάτωθι μεταβλητές:

- ID: η μεταβλητή κλειδί, που υποδεικνύει μία συγκεκριμένη πτήση
- Timestamp: η μεταβλητή, η οποία μας υποδεικνύει την χρονική στιγμή κατά την οποία πάρθηκαν οι μετρήσεις και αναφέρεται σε epochs, δηλαδή μετρήσεις σε milliseconds από 1/1/1970.
- Longitude: η μεταβλητή που υποδεικνύει το γεωγραφικό μήκος τη στιγμή της μέτρησης, σε δεκαδικές μοίρες.
- Latitude: η μεταβλητή που υποδεικνύει το γεωγραφικό πλάτος τη στιγμή της μέτρησης, σε δεκαδικές μοίρες.
- Altitude: η μεταβλητή που υποδεικνύει το γεωγραφικό ύψος τη στιγμή της μέτρησης σε πόδια.
- Speed: η μεταβλητή που υποδεικνύει την ταχύτητα του αεροσκάφους τη στιγμή της μέτρησης, σε κόμβους.
- Heading: η μεταβλητή που υποδεικνύει το αζιμούθιο της κίνησης στο έδαφος, σε μοίρες.

Στο σχήμα που ακολουθεί, παρουσιάζουμε τις τροχιές στον χάρτη, σε δύο διαστάσεις.



Σχήμα 34: Πτήσεις μεταξύ Μαδρίτης-Βαρκελώνης σε δύο διαστάσεις

6.2 Εφαρμογή

6.2.1 Επαναδειγματοληψία *(Resampling)*

Λόγω της φύσης των δεδομένων, ότι δηλαδή προέρχονται από μετρήσεις GPS και δεν υπάρχει σταθερή δειγματοληψία στα σημεία, χρειάστηκε να κάνουμε ορισμένες μετατροπές προκειμένου να τα χρησιμοποιήσουμε στο μοντέλο το οποίο θα χτίσουμε. Ξεκινάμε κανονικοποιώντας την μεταβλητή timestamp ως εξής

$$timestamp'_{i,j} = \frac{timestamp_{i,j} - min_j\left(timestamp_{i,j}\right)}{max_j\left(timestamp_{i,j}\right) - min_j\left(timestamp_{i,j}\right)}$$

Όπου i, j αναφέρεται $i - οστή$, της j τροχιάς.

Ενώ στη συνέχεια χτίζουμε τον παρακάτω αλγόριθμο

Αλγόριθμος 1: Επαναδειγματοληψία

1. **Είσοδος:** Σύνολο ακατέργαστων δεδομένων
2. Ορισμός του επιθυμητού μήκους των μεταβλητών, μέσω της συνάρτησης linspace.
3. Δημιουργία ενός κενού πίνακα.
4. Για κάθε μία από τις πτήσεις:
    a. Εφαρμόζουμε μέθοδο γραμμικής παρεμβολής των σημείων, δημιουργώντας διανύσματα μήκους, όπως ορίσαμε προηγουμένως.
    b. Αφαιρούμε τις NaN τιμές που δημιουργούνται στην αρχή και στο τέλος του διανύσματος, από την παραπάνω μέθοδο, αντικαθιστώντας τα με τα επόμενα σημεία ή προηγούμενα σημεία αντιστοίχως.
    c. Γεμίζουμε τον κενό πίνακα που είχαμε δημιουργήσει.
5. **Έξοδος:** Πίνακας με τις μεταβλητές και αντίστοιχες γραμμές, όσες ορίσαμε στην αρχή του αλγορίθμου.

Πριν, προχωρήσουμε παρακάτω, καλό θα ήταν να ορίσουμε την έννοια της παρεμβολής, στην οποία βασιστήκαμε για την επαναδειγματοληψία. (wikipedia, χ.χ.)

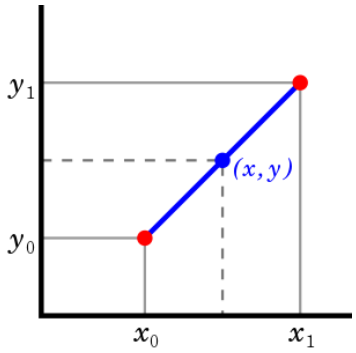
Ορισμός Παρεμβολής

Δεδομένης μίας σειράς μετρήσεων μίας ποσότητας για συγκεκριμένες τιμές της ανεξάρτητης μεταβλητής, ζητείται να βρεθεί μία συνάρτηση, η οποία να έχει τιμή ίση με τις μετρήσεις στις θέσεις της ανεξάρτητης μεταβλητής. Η συνάρτηση μπορεί να χρησιμοποιηθεί για τον υπολογισμό της ποσότητας αυτής σε άλλες θέσεις στις οποίες δεν υπάρχει μέτρηση.

Ενώ θα αναφέρουμε και στην μέθοδο την οποία χρησιμοποιήσαμε στον αλγόριθμο.

Γραμμική Παρεμβολή

Αν θεωρηθεί ότι για μία συνάρτηση είναι γνωστές ορισμένες μόνο τιμές στα σημεία x_i, του πεδίου ορισμού της, η γραμμική παρεμβολή είναι μία απλή μέθοδος με την οποία επιτυγχάνεται ο προσδιορισμός της τιμής μιας συνάρτησης για ένα τυχαίο σημείο x.



Σχήμα 35: Γραμμική Παρεμβολή

Θεωρώντας ότι δύο γνωστά σημεία έχουν τις συντεταγμένες (x_0, y_0) και (x_1,y_1), η γραμμική παρεμβολή βασίζεται στην παραδοχή ότι τα σημεία αυτά ενώνονται με μία ευθεία. Άρα, για ένα τυχαίο σημείο x μεταξύ των x_0 και x_1, η τεταγμένη του προκύπτει από την εξής σχέση:

$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0}$$

$$y = y_0 + (y_1 - y_0)\frac{x - x_0}{x_1 - x_0}$$

6.2.2 Μετατροπή Δεδομένων για Εισαγωγή στα Νευρωνικά Δίκτυα

Εφόσον καταφέραμε να μετατρέψουμε τη δειγματοληψία των δεδομένων σε σταθερή, επόμενο βήμα είναι να μετατρέψουμε τα δεδομένα έτσι, ώστε να είναι ικανά να εισαχθούν στον αλγόριθμο των νευρωνικών δικτύων. (Brownlee, machinelearningmastery, 2016)

Αρχικά, ξεκινάμε κανονικοποιώντας τις μεταβλητές που θα χρησιμοποιήσουμε σαν είσοδο στα νευρωνικά δίκτυα. Αυτό το κάνουμε, διότι οι μεταβλητές μας δεν είναι όλες στην ίδια κλίμακα και τα νευρωνικά είναι πολύ ευαίσθητα σε αυτό.

Στη συνέχεια, δημιουργούμε μία συνάρτηση με σκοπό να προσαρμόσουμε τα δεδομένα μας, ώστε να είναι σε κατάλληλη μορφή για την είσοδο στα δίκτυα.

Αλγόριθμος 2: Δημιουργία Συνάρτησης Create_Dataset

1. **Είσοδος:** Όρισε το Dataset και το Look_Back
2. Δημιούργησε δύο κενούς πίνακες X, Y
3. Για κάθε i στο μήκος του Dataset-LookBack-1
   $a = Dataset[i{:}\, i + LookBack)$,0]
   $\beta = Dataset[(i + LookBack)$,0]
   Κάνε append στον πίνακα X το α
   Κάνε append στον πίνακα Y το β
4. **Έξοδος:** Επέστρεψε τον πίνακα X και τον πίνακα Y ως arrays.

Το Look_Back στον παραπάνω αλγόριθμο, δηλώνει το πόσα προηγούμενα σημεία του δείγματος θα χρησιμοποιεί το μοντέλο για να προβλέψει το επόμενο. [10]

Έτσι, χρησιμοποιώντας την παραπάνω συνάρτηση, δημιουργούμε έναν πίνακα X, ο οποίος περιέχει τον αριθμό των σημείων στο χρόνο t και έναν πίνακα Y, ο οποίος περιέχει τον αριθμό των σημείων στο αμέσως επόμενο χρονικό σημείο $t + 1$.

Πριν προχωρήσουμε στη διαμόρφωση της αρχιτεκτονικής των νευρωνικών δικτύων, χωρίζουμε το δείγμα σε δύο υπό-δείγματα, χρησιμοποιώντας το ένα για εκπαίδευση του δικτύου και το δεύτερο για την πρόβλεψη του μοντέλου μας. Για την ακρίβεια, χρησιμοποιήσαμε 1277 πτήσεις για εκπαίδευση και δοκιμάσαμε να προβλέψουμε άλλες 119.

6.2.3 Αρχιτεκτονική Νευρωνικών Δικτύων

Χρησιμοποιώντας την βιβλιοθήκη Keras, όπως έχουμε αναφέρει, και έπειτα από διάφορες δοκιμές που έγιναν, καταλήξαμε στην παρακάτω αρχιτεκτονική για τα νευρωνικά δίκτυα, η οποία αποτελείται από δύο κρυμμένα στρώματα, ένα LSTM με 32 νευρώνες και ένα πλήρως συνδεδεμένο στρώμα με 16 νευρώνες και συνάρτηση ενεργοποίησης τη ReLU. Στο στρώμα εξόδου επιλέχθηκε η Σιγμοειδής συνάρτηση ενεργοποίησης, εφόσον και τα δεδομένα μας είχαν κανονικοποιηθεί.

Αλγόριθμος 3: Αρχιτεκτονική Νευρωνικών Δικτύων

1. **Είσοδος:** Διάνυσμα μήκους 5, όσες και οι μεταβλητές του δείγματος μας
2. **Κρυμμένα Στρώματα**
   a. **Επίπεδο 1:** LSTM/GRU/SimpleRNN στρώμα με 32 νευρώνες
   b. **Επίπεδο 2:** Πλήρως συνδεδεμένο στρώμα (fully connected) με 16 νευρώνες. Ως συνάρτηση ενεργοποίησης χρησιμοποιούμε τη ReLU.
3. **Έξοδος:** Στρώμα εξόδου με 3 νευρώνες*(όσες και οι μεταβλητές που θέλουμε να προβλέψουμε)* και σιγμοειδή συνάρτηση ενεργοποίησης.

6.2.4 Εκπαίδευση του Νευρωνικού Δικτύου

Για την εκπαίδευση του μοντέλου μας, βασιστήκαμε στη συνάρτηση κόστους ελαχίστων τετραγώνων *(MSE)*, ενώ σαν αλγόριθμος βελτιστοποίησης επιλέχθηκε ο Adam

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

Όπου $\hat{Y}$, είναι το διάνυσμα των προβλέψεων και Y_i το διάνυσμα των πραγματικών παρατηρήσεων μας.

Κατά την εκπαίδευση του μοντέλου μας, χρησιμοποιήσαμε τα δεδομένα όπως τα είχαμε χωρίσει σε Train και Test σετ προηγουμένως, ενώ το εκπαιδεύσαμε για 50 εποχές *(epochs)*, χρησιμοποιώντας μέγεθος δείγματος *(batch size)* κάθε εποχή, 512, διότι τα δεδομένα μας ήταν πολλά και δεν ήταν δυνατό να διαβάζονται όλα σε κάθε εποχή της εκπαίδευσης.

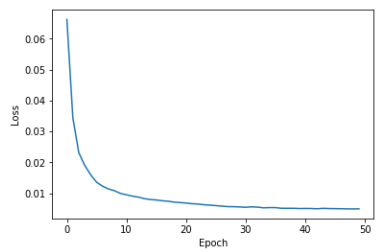
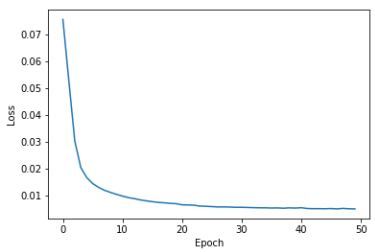
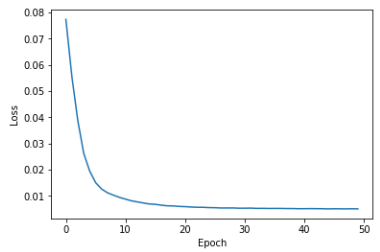
6.3 Αποτελέσματα

Σε αυτό το κομμάτι, θα παρουσιάσουμε τα αποτελέσματα, τα οποία προέκυψαν από την υλοποίηση του μοντέλου. Έχουμε δοκιμάσει να συγκρίνουμε τρία είδη επανατροφοδοτούμενων δικτύων, τα Simple RNN, τα LSTM και τα GRU. Ενώ έχουμε κάνει δοκιμές για διαφορετικά μεγέθη δειγματοληψίας για να παρουσιάσουμε το κατά πόσο ή όχι βελτιώνονται τα αποτελέσματα.

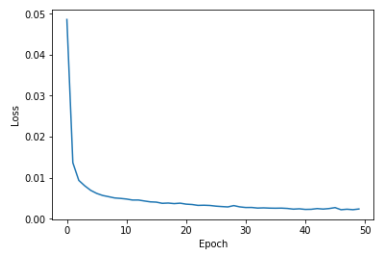
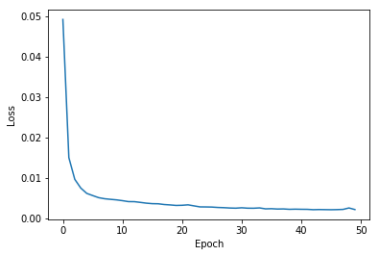
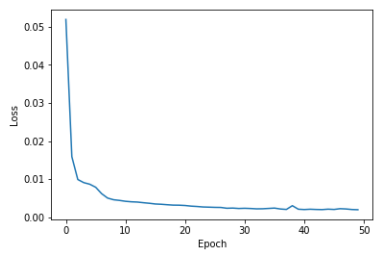
6.3.1 Εκπαίδευση

Στα ακόλουθα γραφήματα, παρατηρούμε πως μειώνεται το σφάλμα εκπαίδευσης σε κάθε μία από τις προαναφερθέντες περιπτώσεις.

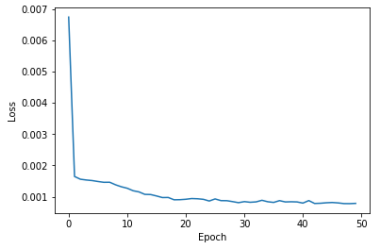
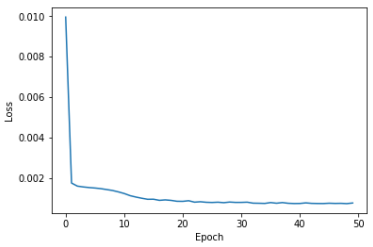
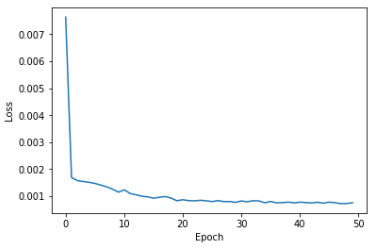
Για μέγεθος δειγματοληψίας 20 ***(ανά περίπου 330 δευτερόλεπτα)***

Simple RNN	LSTM	GRU
		

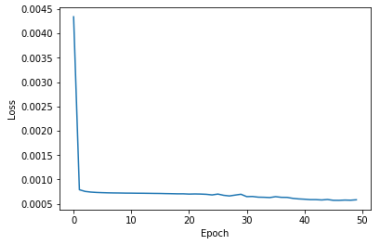
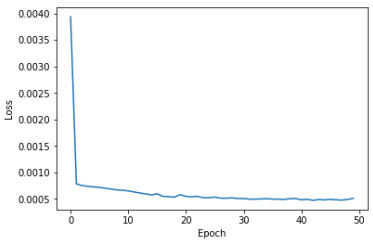
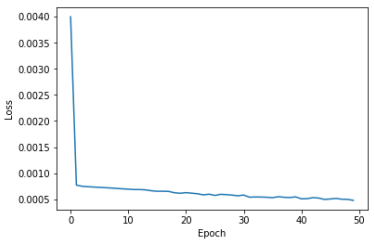
Για μέγεθος δειγματοληψίας 50 ***(ανά περίπου 132 δευτερόλεπτα)***

Simple RNN	LSTM	GRU
		

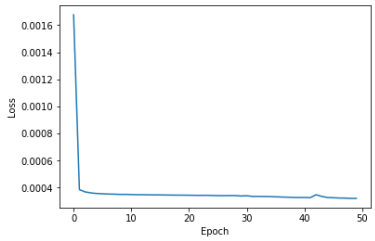
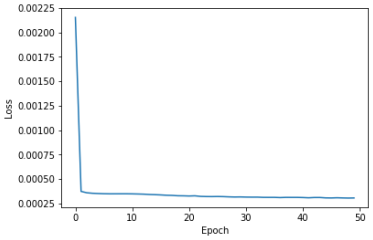
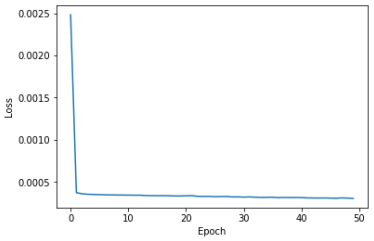
Για μέγεθος δειγματοληψίας 300 ***(ανά περίπου 22 δευτερόλεπτα)***

Simple RNN	LSTM	GRU
		

Για μέγεθος δειγματοληψίας 651 ***(ανά περίπου 10 δευτερόλεπτα)***

Simple RNN	LSTM	GRU
		

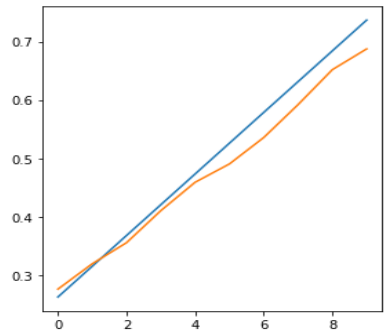
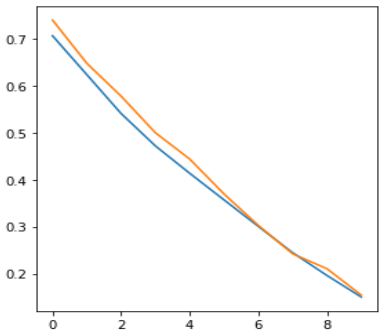
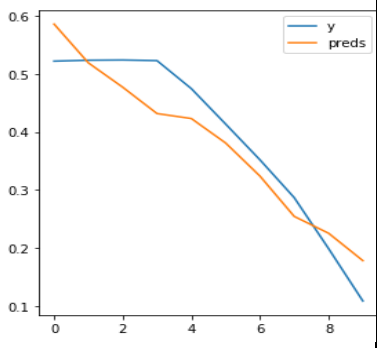
Για μέγεθος δειγματοληψίας 1354 ***(ανά περίπου 5 δευτερόλεπτα)***

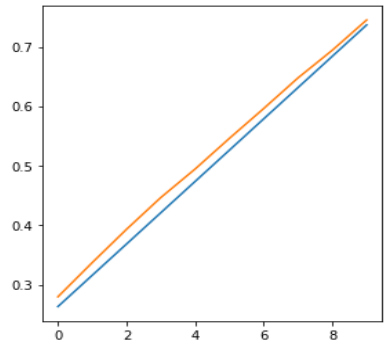
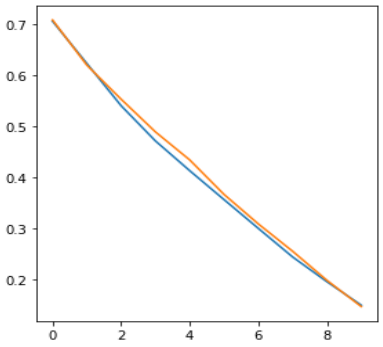
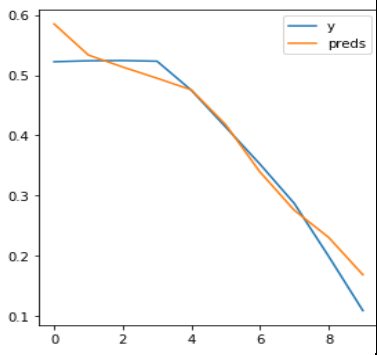
Simple RNN	LSTM	GRU
		

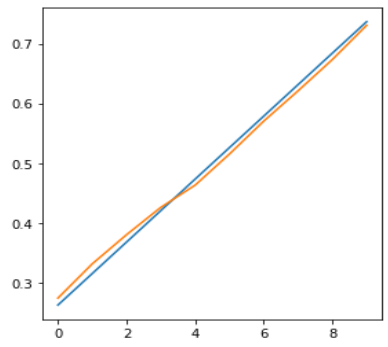
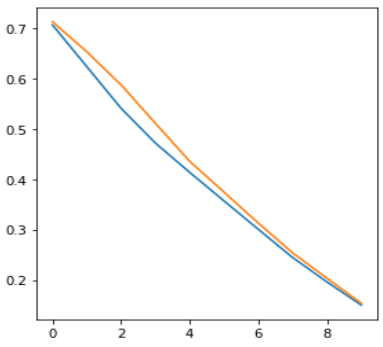
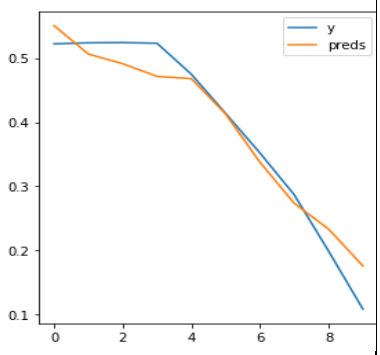
6.3.2 Πρόβλεψη στην επόμενη χρονική στιγμή

Στα ακόλουθα γραφήματα, παρατηρούμε τις προβλέψεις σε κάθε μία περίπτωση.

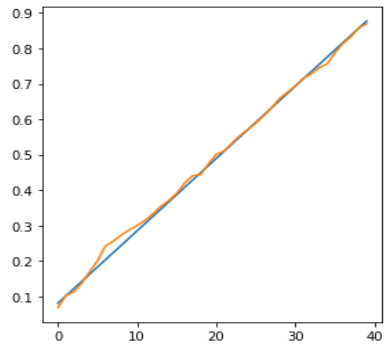
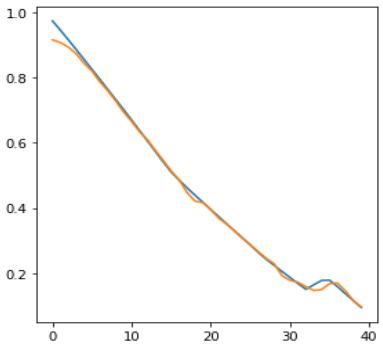
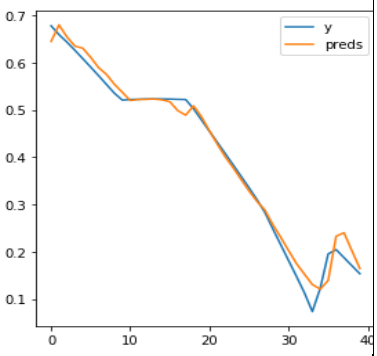
Για μέγεθος δειγματοληψίας 20

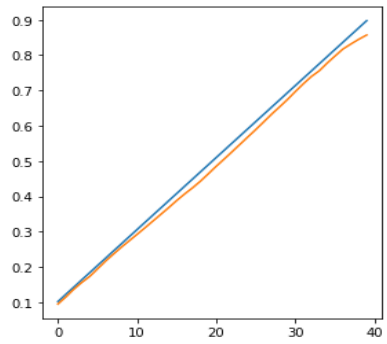
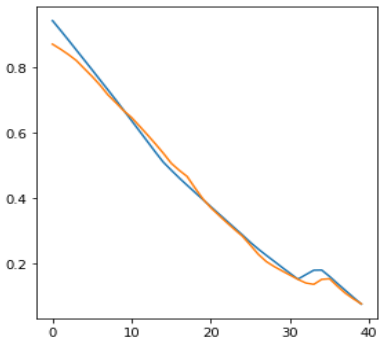
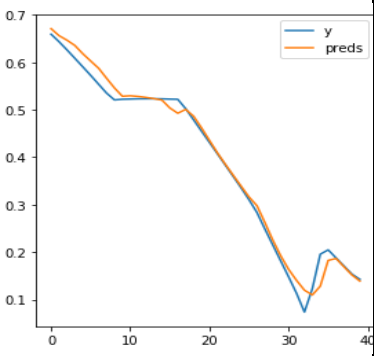
Simple RNN		
Longitude	**Latitude**	**Altitude**
		

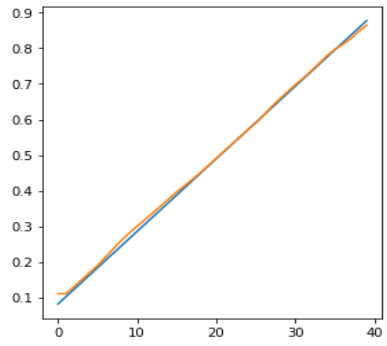
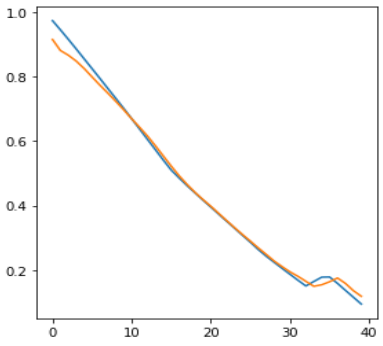
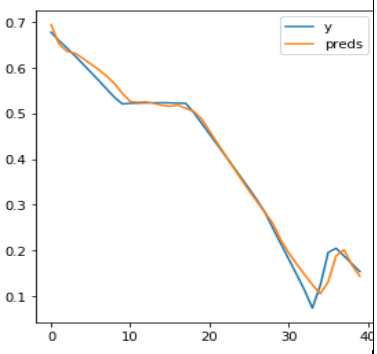
LSTM		
Longitude	**Latitude**	**Altitude**
		

GRU		
Longitude	**Latitude**	**Altitude**
		

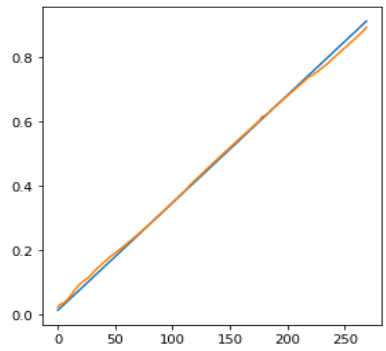
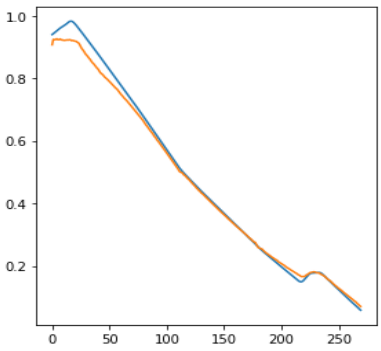
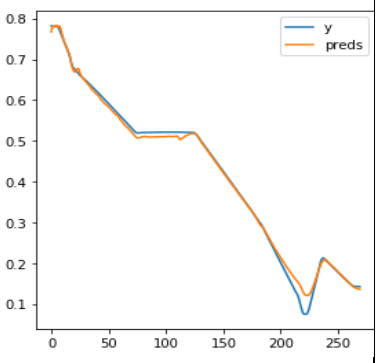
Για μέγεθος δειγματοληψίας 50

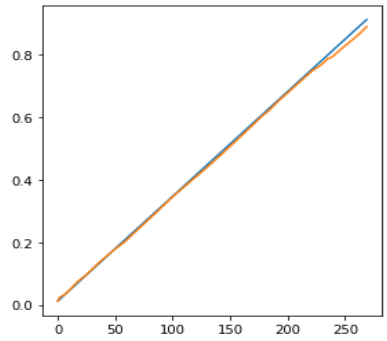
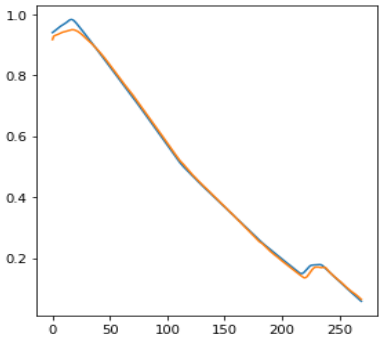
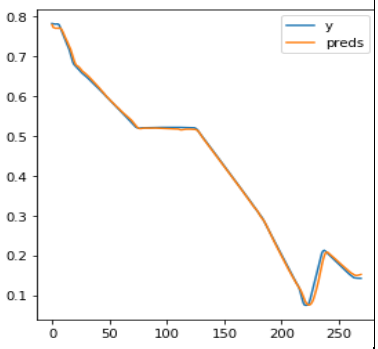
Simple RNN		
Longitude	**Latitude**	**Altitude**
		

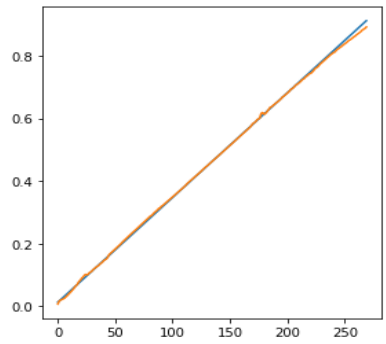
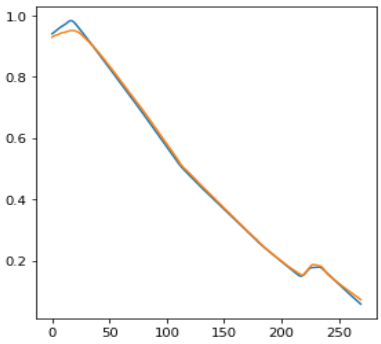
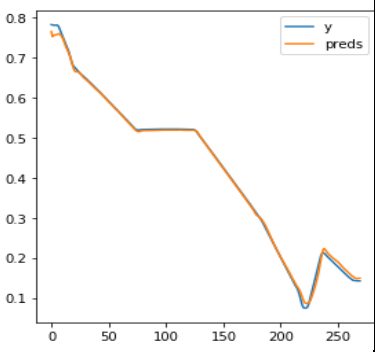
LSTM		
Longitude	**Latitude**	**Altitude**
		

GRU		
Longitude	**Latitude**	**Altitude**
		

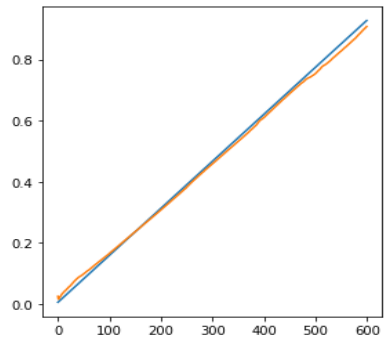
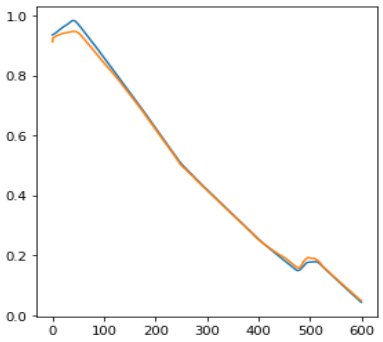
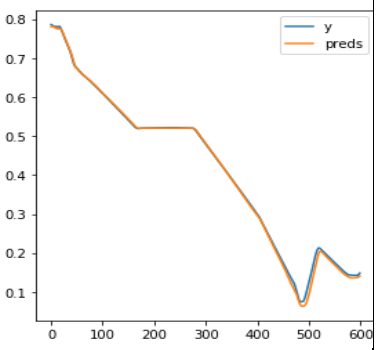
Για μέγεθος δειγματοληψίας 300

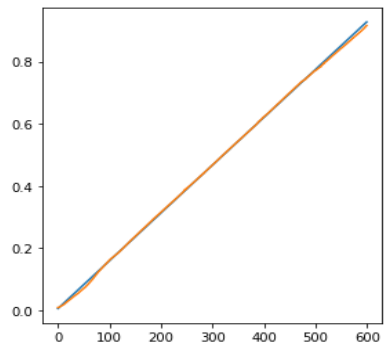
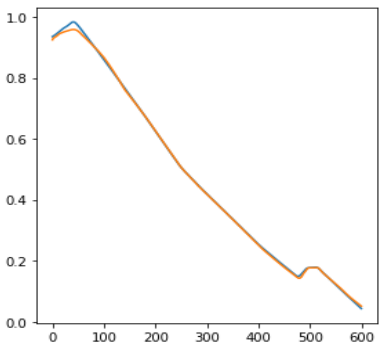
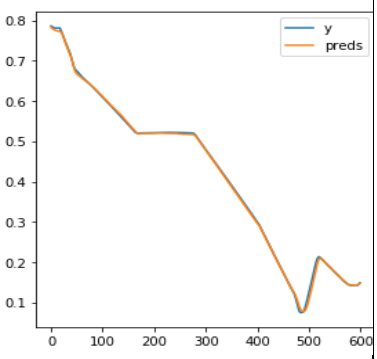
Simple RNN		
Longitude	**Latitude**	**Altitude**
		

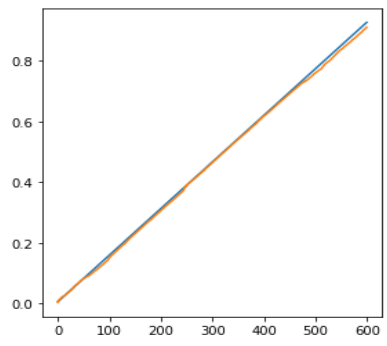
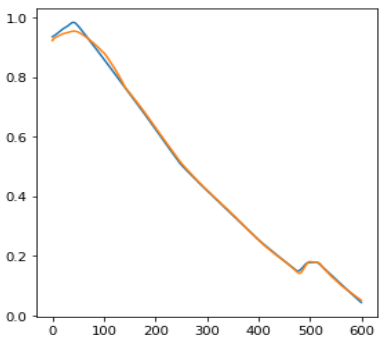
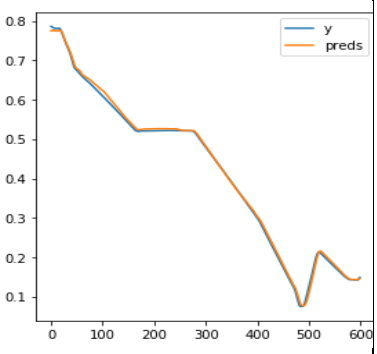
LSTM		
Longitude	**Latitude**	**Altitude**
		

GRU		
Longitude	**Latitude**	**Altitude**
		

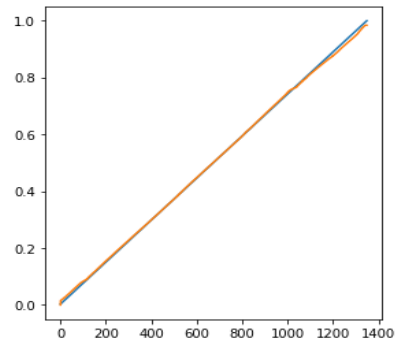
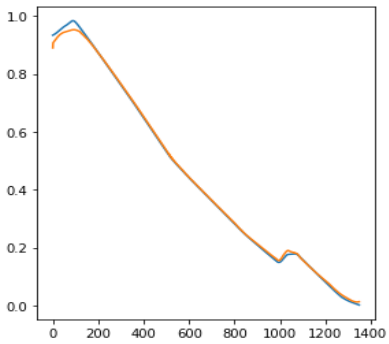
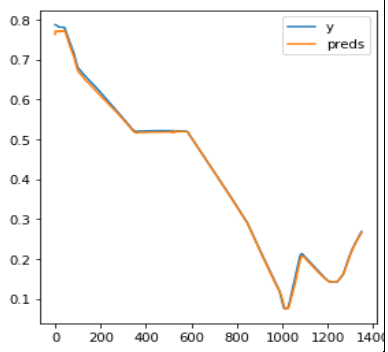
Για μέγεθος δειγματοληψίας 651

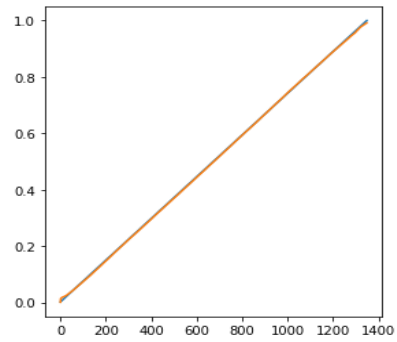
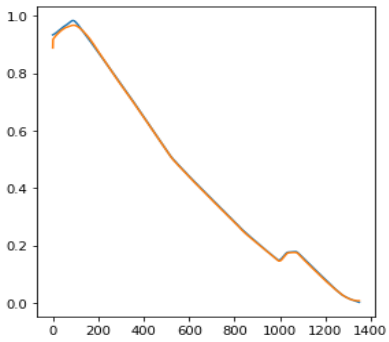
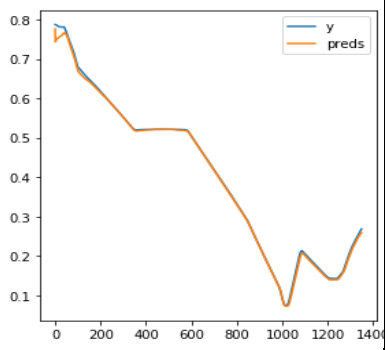
Simple RNN		
Longitude	**Latitude**	**Altitude**
		

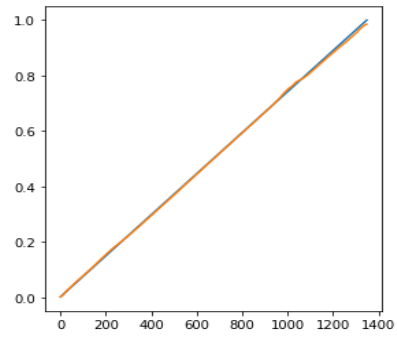
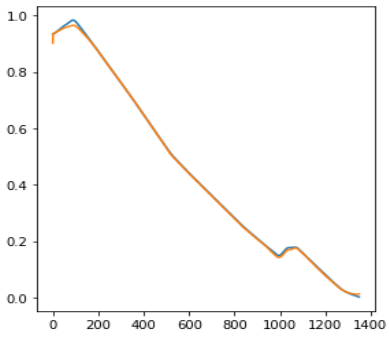
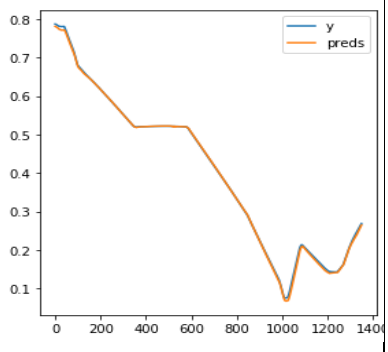
LSTM		
Longitude	**Latitude**	**Altitude**
		

GRU		
Longitude	**Latitude**	**Altitude**
		

Για μέγεθος δειγματοληψίας 1354

Simple RNN		
Longitude	**Latitude**	**Altitude**
		

LSTM		
Longitude	**Latitude**	**Altitude**
		

GRU		
Longitude	**Latitude**	**Altitude**
		

6.3.3 Σύγκριση Αποτελεσμάτων

Σε αυτό το κομμάτι, θα παρουσιάσουμε τους πίνακες με τις συγκρίσεις των παραπάνω αποτελεσμάτων, αναφορικά τόσο με το μέσο τετραγωνικό σφάλμα μέτρησης, όσο και με τους χρόνους εκτέλεσης σε κάθε μία από τις περιπτώσεις ξεχωριστά.

MSE για το Altitude					
	20	**50**	**300**	**651**	**1354**
LSTM	0,001448	0,001036	0,000923	0,000558	0,000502
GRU	0,003871	0,001529	0,000932	0,000824	0,000532
RNN	0,003565	0,001133	0,001059	0,000968	0,000637

MSE για το Longitude και Latitude					
	20	**50**	**300**	**651**	**1354**
LSTM	0,001483	0,001298	0,000901	0,000649	0,000525
GRU	0,003301	0,001865	0,001031	0,000842	0,000543
RNN	0,002839	0,001680	0,001147	0,001067	0,000694

ETA *(in seconds)*					
	20	**50**	**300**	**651**	**1354**
LSTM	28,67	76,91	454,39	533,33	928,19
GRU	24,09	68,15	390,09	482,83	898,95
RNN	10,26	29,36	156,75	222,91	430,24

ETA *(in minutes)*					
	20	**50**	**300**	**651**	**1354**
LSTM	0,48	1,28	7,57	8,89	15,47
GRU	0,40	1,14	6,50	8,05	14,98
RNN	0,17	0,49	2,61	3,72	7,17

6.3.4 Πρόβλεψη σε περισσότερες χρονικές στιγμές

Μέχρι τώρα, δοκιμάσαμε και πήραμε αρκετά ικανοποιητικά αποτελέσματα, στην προσπάθεια μας να χρησιμοποιήσουμε τα μοντέλα για την πρόβλεψη στην αμέσως επόμενη χρονική στιγμή. Σε αυτό το κομμάτι της εργασίας, θα προσπαθήσουμε να δούμε τι συμβαίνει όταν δοκιμάζουμε σε περισσότερες χρονικές στιγμές.

Για να το κάνουμε αυτό, πρέπει αρχικά να ξανατρέξουμε τα μοντέλα, όπως πριν, με τη μόνη διαφορά, ότι τώρα θα θέσουμε σαν έξοδο από αυτά και τις πέντε μεταβλητές εισόδου. Ο σκοπός είναι να ξαναδίνουμε κάθε φορά στο μοντέλο τις τιμές που προβλέπουμε.

Έτσι προκειμένου να κάνουμε τις προβλέψεις κάθε φορά, χρειαζόμαστε έναν πίνακα διαστάσεων (1,5,5), όπου η πρώτη διάσταση είναι το batch size, η δεύτερη είναι το look back και η τρίτη ο αριθμός των μεταβλητών.

Παρουσιάζουμε ένα παράδειγμα για το πώς θα είναι οι πίνακες που θα δημιουργούνται με κάθε επανάληψη στο μοντέλο.

	f1	f2	f3	f4	f5
t1	1	2	3	4	5
t2	6	7	8	9	10
t3	11	12	13	14	15
t4	16	17	18	19	20
t5	21	22	23	24	25

Όταν δίνουμε αυτό σαν είσοδο στο μοντέλο, του ζητάμε να προβλέψει την επόμενη χρονική στιγμή $t6$.

	f1	f2	f3	f4	f5
t6	26	27	28	29	30

Έπειτα αυτό το προσθέτουμε στο τέλος της προηγούμενης εισόδου, αφαιρώντας την πρώτη από τις 5 προηγούμενες χρονικές στιγμές.

	f1	f2	f3	f4	f5
t2	6	7	8	9	10
t3	11	12	13	14	15
t4	16	17	18	19	20
t5	21	22	23	24	25
t6	26	27	28	29	30

Στη συνέχεια προσπαθούμε να προβλέψουμε την χρονική στιγμή $t7$.

	f1	f2	f3	f4	f5
t7	31	32	33	34	35

Και επαναλαμβάνουμε το ίδιο μέχρι το τέλος.

Αυτό το πραγματοποιούμε με τον παρακάτω αλγόριθμο.

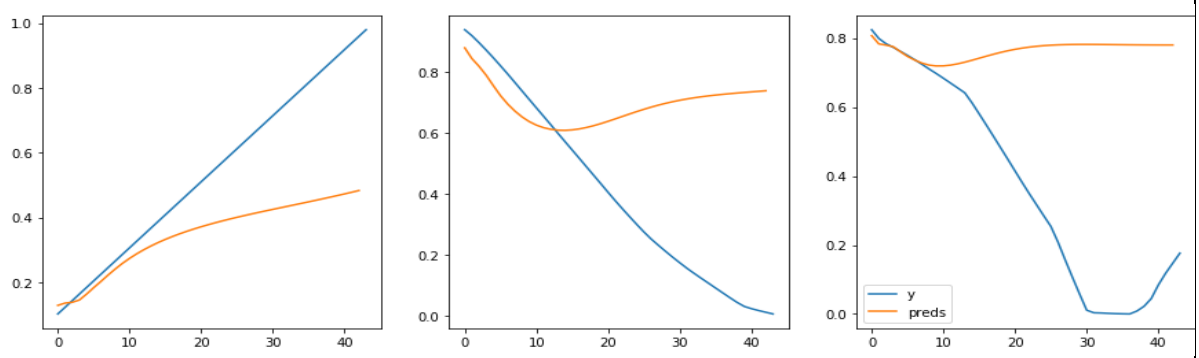
Αλγόριθμος 4: Πίνακας πρόβλεψης πολλαπλών χρονικών στιγμών

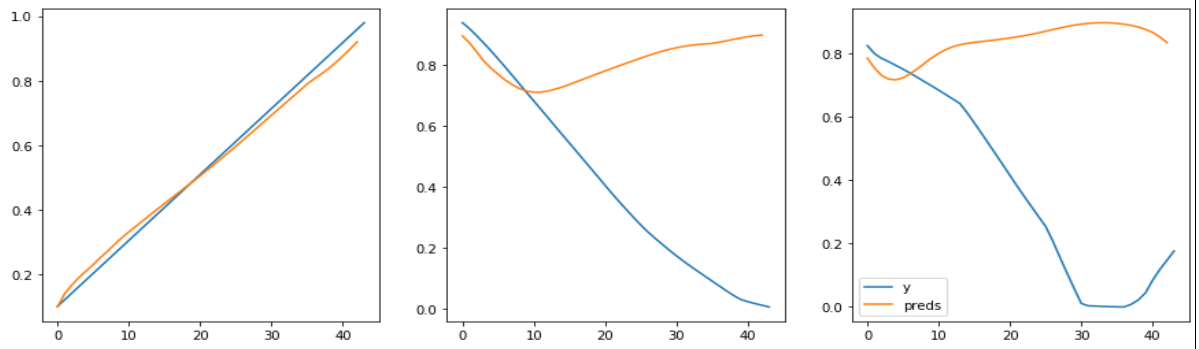
1. Δημιούργησε έναν κενό πίνακα predictions
2. Όρισε την πρώτη χρονική στιγμή
3. Για κάθε _ στο μήκος των προβλέψεων του μοντέλου y_ts [(11)]
   Προέβλεψε την επόμενη χρονική στιγμή
   Ένωσε με τις 4 προηγούμενες χρονικές στιγμές
   Κάνε append στον πίνακα predictions

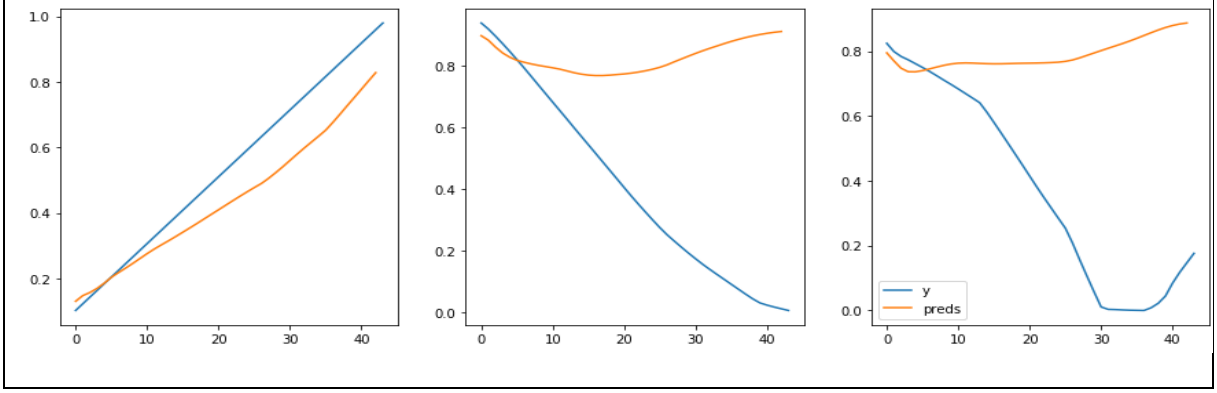
6.3.5 Αποτελέσματα Πρόβλεψης σε περισσότερες χρονικές στιγμές

Για αυτό το κομμάτι της εργασίας, κρατήσαμε τρία μεγέθη δειγματοληψίας, ώστε να είναι πιο συγκρίσιμα τα αποτελέσματα.

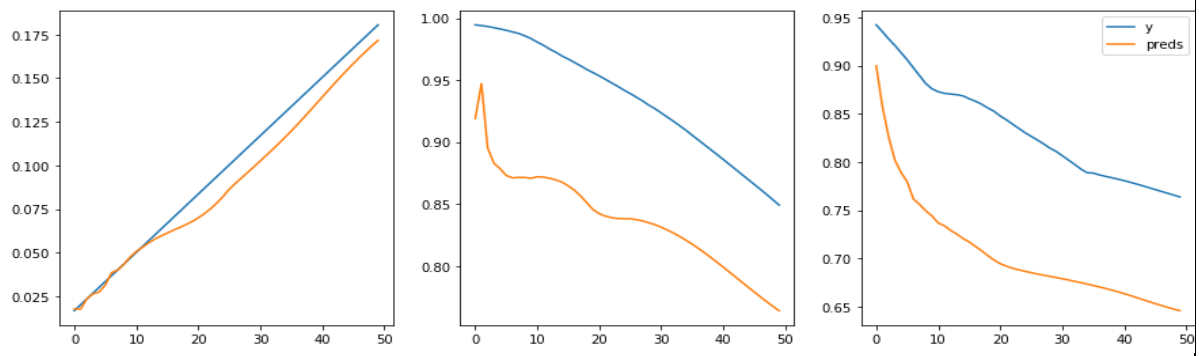
Για μέγεθος δειγματοληψίας 50 ***(ανά περίπου 132 δευτερόλεπτα)***

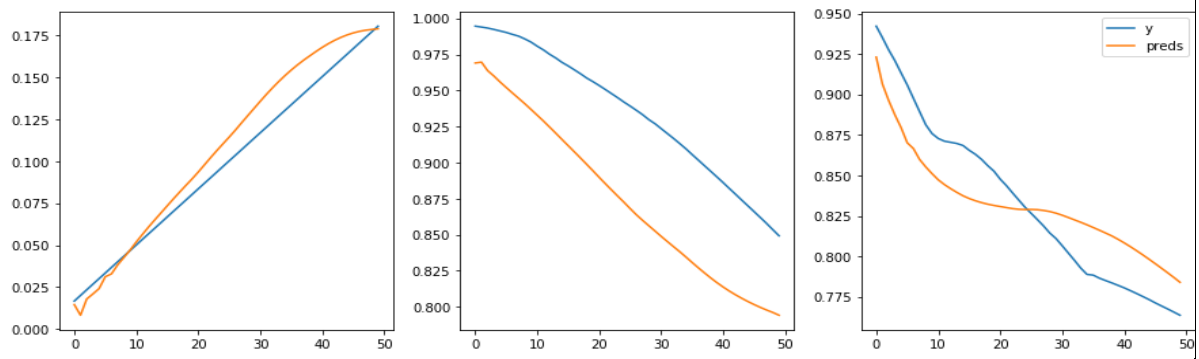
Simple RNN		
Longitude	**Latitude**	**Altitude**
		

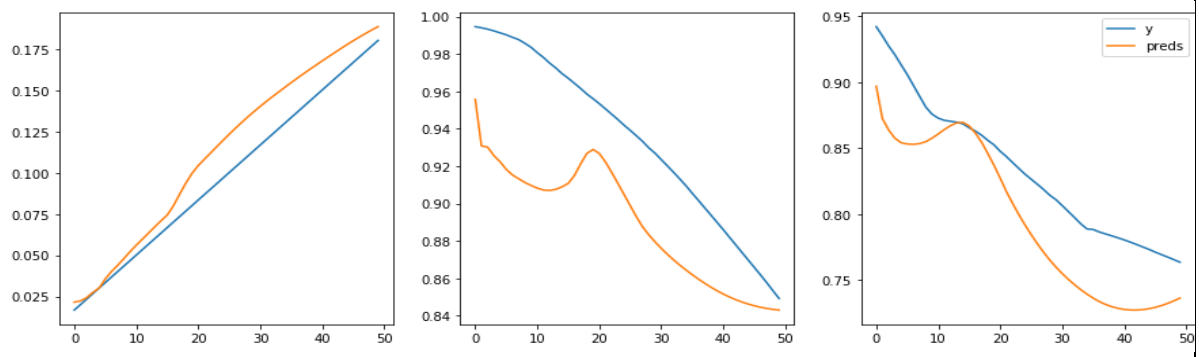
LSTM		
Longitude	**Latitude**	**Altitude**
		

GRU		
Longitude	**Latitude**	**Altitude**
		

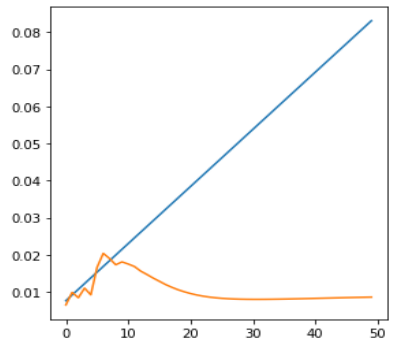
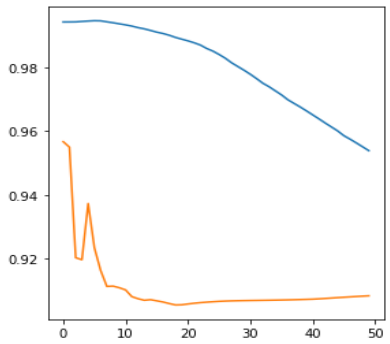
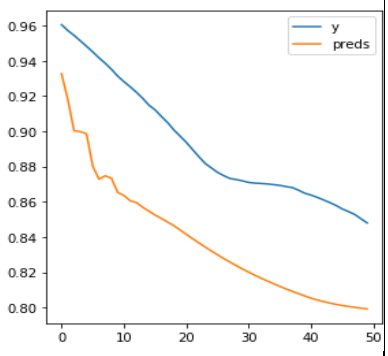
Για μέγεθος δειγματοληψίας 300 ***(ανά περίπου 22 δευτερόλεπτα)***

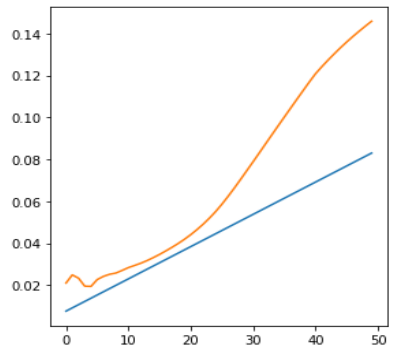
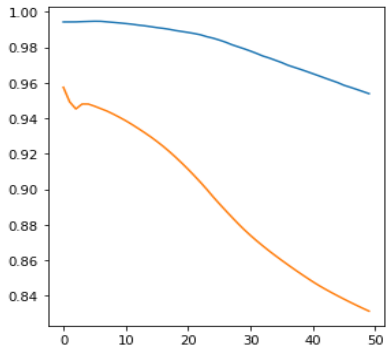
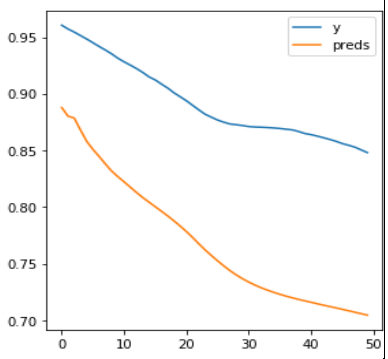
Simple RNN		
Longitude	**Latitude**	**Altitude**
		

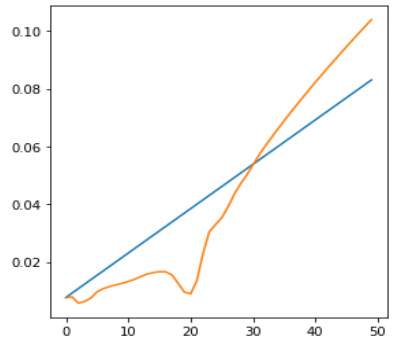
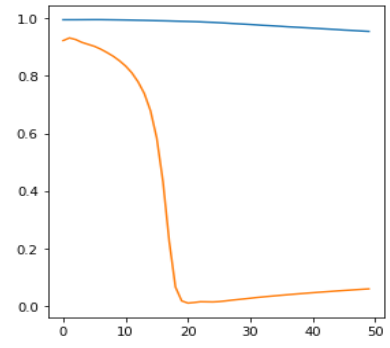
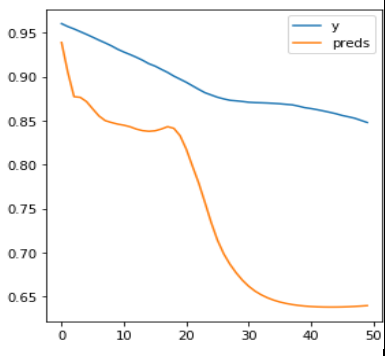
LSTM		
Longitude	**Latitude**	**Altitude**
		

GRU		
Longitude	**Latitude**	**Altitude**
		

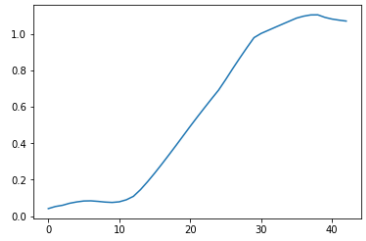
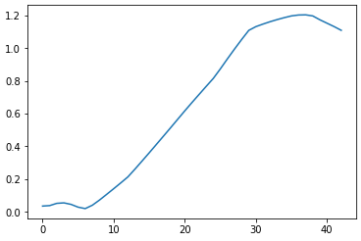
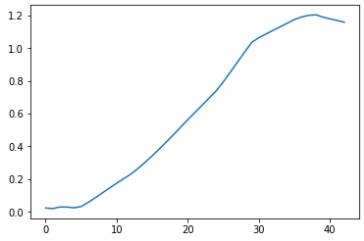
Για μέγεθος δειγματοληψίας 651*(ανά περίπου 10 δευτερόλεπτα)*

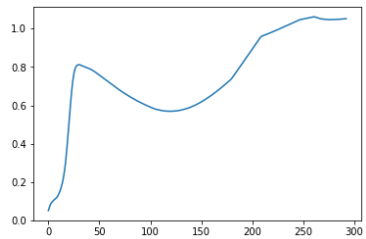
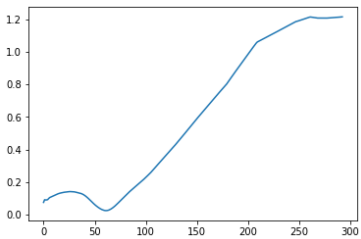
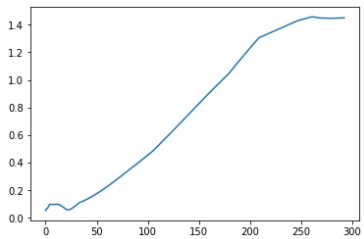
Simple RNN		
Longitude	**Latitude**	**Altitude**
		

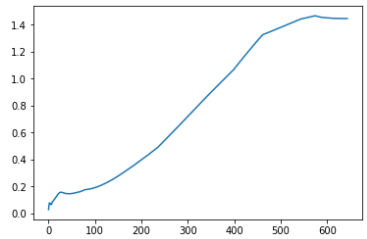
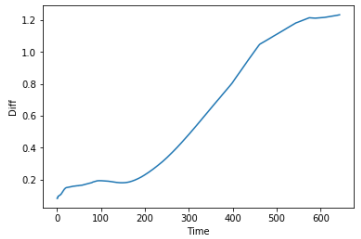
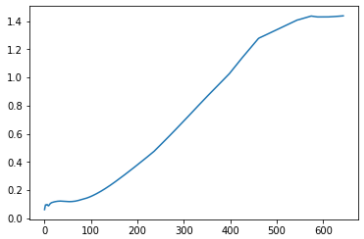
LSTM		
Longitude	**Latitude**	**Altitude**
		

GRU		
Longitude	**Latitude**	**Altitude**
		

Ενώ δημιουργήσαμε τις διαφορές μεταξύ της πραγματικής και της προβλεπόμενης τροχιάς για τις τρεις διαστάσεις σε πολλαπλές στιγμές στο χρόνο, μέσω της ευκλείδειας απόστασης, παρατηρούμε ότι η διαφορά αυτή αυξάνεται συνεχώς.

Δειγματοληψία 50		
Simple RNN	**LSTM**	**GRU**
		

Δειγματοληψία 300		
Simple RNN	**LSTM**	**GRU**
		

Δειγματοληψία 651		
Simple RNN	**LSTM**	**GRU**
		

6.3.6 Συμπεράσματα

Ξεκινήσαμε την υλοποίηση της εργασίας με σκοπό να προβλέψουμε το επόμενο σημείο χρονικά σε μία τροχιά με τη χρήση τριών διαφορετικών μοντέλων νευρωνικών δικτύων Simple RNN, LSTM και GRU. Επιπλέον, κάναμε δοκιμές σε διαφορετικά μεγέθη δειγματοληψίας προκειμένου να δούμε το πώς συμπεριφέρονται τα μοντέλα αυτά.

Τα αποτελέσματα, όπως φαίνονται στην προηγούμενη ενότητα, μας έδειξαν ότι και τα τρία μοντέλα συμπεριφέρονται εξίσου καλά σε αυτή την προσπάθεια. Με βάση το κριτήριο MSE φαίνεται ότι το μοντέλο του LSTM συμπεριφέρεται λίγο καλύτερα από τα άλλα δύο. Όμως, με βάση τους χρόνους εκτέλεσης δείχνει το RNN να αποδίδει καλύτερα, αφού φαίνεται ότι εκπαιδεύεται αρκετά πιο γρήγορα σε σχέση με τα άλλα δύο, για την ακρίβεια χρειάζεται σχεδόν το μισό χρόνο.

Στη συνέχεια, δοκιμάσαμε να δούμε το πώς τα προαναφερθέντα μοντέλα αντιδράνε όταν προσπαθούμε να προβλέψουμε σε περισσότερες χρονικές στιγμές στο μέλλον.

Σε αυτό το κομμάτι, σε αντίθεση με την προηγούμενη προσπάθεια, παρατηρούμε ότι δεν ανταποκρίνονται ικανοποιητικά τα μοντέλα αυτά. Όσο περισσότερο προχωράμε τα time steps, όπως είναι λογικό, τόσο περισσότερο χειροτερεύουν οι προβλέψεις και μετά από ένα χρονικό σημείο, ανάλογα και το μέγεθος της δειγματοληψίας κάθε φορά, φαίνεται ότι χάνουν τελείως την προβλεπτική τους ικανότητα, όπως φαίνεται και στα γραφήματα. Τέλος, να αναφέρουμε, ότι το απλό RNN μοντέλο, δείχνει να έχει την μεγαλύτερη δυσκολία, όσον αφορά αυτό το κομμάτι, ενώ τα άλλα δύο μοντέλα όπως και στην προηγούμενη περίπτωση παρουσιάζουν παρόμοια αποτελέσματα.

Παράρτημα Α

(1) perceptron's bias, $b \equiv -threshold$

(2) Η NAND gate (negative-AND) είναι μια λογική πύλη που παράγει μια έξοδο η οποία είναι ψευδής μόνο αν όλες οι είσοδοι της είναι αληθείς.

(3) Το σ, μερικές φορές καλείται λογιστική συνάρτηση και ο τύπος αυτός των νευρώνων καλείται Λογιστικοί Νευρώνες (Logistic Neurons).

(4) Στην πραγματικότητα, όταν $w * x + b = 0$, το perceptron, δίνει έξοδο 0, ενώ η βηματική συνάρτηση δίνει 1. Έτσι πιο αυστηρά, θα έπρεπε να τροποποιήσουμε τη βηματική συνάρτηση σε αυτό το κομμάτι.

(5) Στη βιβλιογραφία, συχνά όταν αναφέρεται στην τιμή που ελαχιστοποιεί τη συνάρτηση κόστους, χρησιμοποιείται ο αστερίσκος *, για παράδειγμα, $x^* = \arg\min C(x)$.

(6) ∇ είναι ο τελεστής κλίσης $\left[\frac{\partial}{\partial v_1}, \frac{\partial}{\partial v_2}\right]^T$

$\nabla C(\boldsymbol{v})$ είναι το διάνυσμα κλίσης της συνάρτησης κόστους $\nabla C(\boldsymbol{v}) = \left[\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2}\right]^T$

$\Delta v = (\Delta v_1, \Delta v_2)^T$

(7) Ως δ^l θα εννοούμε το διάνυσμα των σφαλμάτων του l-οστού στρώματος.

(8) Το σύμβολο ⊙ αναφέρεται στο γινόμενο στοιχείο προς στοιχείο δύο διανυσμάτων ίδιων διαστάσεων και ονομάζεται Hadamard γινόμενο. Για παράδειγμα

$$\begin{bmatrix}1\\2\end{bmatrix} \odot \begin{bmatrix}3\\4\end{bmatrix} = \begin{bmatrix}1*3\\2*4\end{bmatrix} = \begin{bmatrix}3\\8\end{bmatrix}$$

(9) Markov chain: Μια Μαρκοβιανή Αλυσίδα είναι μια ακολουθία τυχαίων μεταβλητών $X_1, X_2, \dots X_n$ για την οποία ισχύει η Μαρκοβιανή Ιδιότητα, δηλαδή με δεδομένη την παρούσα κατάσταση, οι παλαιότερες και οι μελλοντικές καταστάσεις είναι ανεξάρτητες.

$$P(X_{n+1} = x | X_1 = x_1, X_2 = x_2, \dots, X_n = x_n) = P(X_{n+1} = x | X_n = x_n)$$

(10) Look_Back: Για παράδειγμα, εάν έχουμε μία ακολουθία 1,2,3,4,5 και θέλουμε να προβλέψουμε το 6, χρησιμοποιώντας Look_Back = 3, ο αλγόριθμος θα μελετάει τα τρία προηγούμενα σημεία για να προβλέψει το 6, δηλαδή τα 3,4 και 5.

(11) Η κάτω παύλα _, χρησιμοποιείται στην python, προκειμένου να κρατηθεί το τελευταίο αποτέλεσμα σε μία επανάληψη, σε αυτή.

Παράρτημα Β *(Βασικοί Αλγόριθμοι)*

Βοηθητικές Συναρτήσεις

```
def create_dataset(dataset, look_back=1):
dataX, dataY = [], []
for i in range(len(dataset)-look_back-1):
a = dataset[i:(i+look_back), 0]
dataX.append(a)
dataY.append(dataset[i + look_back, 0])
return np.array(dataX), np.array(dataY)

def plot(y1,y2,y3,p1,p2,p3):
plt.figure(figsize=(15,5))
ax = plt.subplot(131)
ax.plot(y1, label='y')
ax.plot(p1, label='preds')
ax = plt.subplot(132)
ax.plot(y2, label='y')
ax.plot(p2, label='preds')
ax = plt.subplot(133)
ax.plot(y3, label='y')
ax.plot(p3, label='preds')
ax.legend()

class TimeHistory(keras.callbacks.Callback):
def on_train_begin(self, logs={}):
self.times = []
def on_epoch_begin(self, batch, logs={}):
self.epoch_time_start = time.time()
def on_epoch_end(self, batch, logs={}):
self.times.append(time.time() - self.epoch_time_start)
```

Αλγόριθμος Επαναδειγματοληψίας

```
flights = data.id.unique()
count_files = 0
planes=[]
for i,index in enumerate(flights):
if count_files == 0:
planes.append(data[data['id'] == flights[i]])
count_files += 1
else:
planes.append(data[data['id'] == flights[i]])
count_files += 1
```

```
for plane in planes:
plane['timestamp'] = (plane['timestamp'] - plane['timestamp'].min()) / (plane['timestamp'].max() - plane['timestamp'].min())

variables = ['longitude', 'latitude', 'altitude', 'speed','heading']
target_time  = np.linspace(0, 1, 300)
total_df = pd.DataFrame(columns=plane1.columns)

for plane in planes:
rel_time = plane['timestamp'].values - plane['timestamp'].values[0]
df = pd.DataFrame(columns=plane.columns)
for var in variables:
# interpolation
interp = interp1d(rel_time, plane[var], kind='linear', fill_value='extrapolate')
predicted = interp(target_time)
# φτιαχνω τα nan:
ind = np.where(~np.isnan(predicted))[0]
first, last = ind[0], ind[-1]
predicted[:first] = predicted[first]
predicted[last + 1:] = predicted[last]
df[var] = predicted
# γεμιζω την στηλη με τα id
df['id'] = plane['id'].values[0]
# γεμιζω τη στηλη με τα 'timestamp'
df['timestamp'] = target_time
# τα προσθετω στο total_df:
total_df = pd.concat([total_df, df])
```

Διαμόρφωση των inputs για το training και test σετ

```
look_back = 5
tmp = sc.fit_transform(train)
x_lon, y_lon = create_dataset(tmp[:, 1].reshape(-1, 1), look_back=look_back)
x_lat, y_lat = create_dataset(tmp[:, 2].reshape(-1, 1), look_back=look_back)
x_alt, y_alt = create_dataset(tmp[:, 3].reshape(-1, 1), look_back=look_back)
x_spe, y_spe = create_dataset(tmp[:, 4].reshape(-1, 1), look_back=look_back)
x_head, y_head = create_dataset(tmp[:, 5].reshape(-1, 1), look_back=look_back)
x = np.stack([x_lon, x_lat, x_alt, x_spe, x_head], axis=-1)
y = np.stack([y_lon, y_lat, y_alt], axis=-1)

tmp = sc.fit_transform(test)
x_lon, y_lon = create_dataset(tmp[:, 1].reshape(-1, 1), look_back=look_back)
x_lat, y_lat = create_dataset(tmp[:, 2].reshape(-1, 1), look_back=look_back)
x_alt, y_alt = create_dataset(tmp[:, 3].reshape(-1, 1), look_back=look_back)
x_spe, y_spe = create_dataset(tmp[:, 4].reshape(-1, 1), look_back=look_back)
x_head, y_head = create_dataset(tmp[:, 5].reshape(-1, 1), look_back=look_back)
x_lon = x_lon.reshape(-1, look_back, 1)
x_lat = x_lat.reshape(-1, look_back, 1)
x_alt = x_alt.reshape(-1, look_back, 1)
x_spe = x_spe.reshape(-1, look_back, 1)
x_head = x_head.reshape(-1, look_back, 1)
```

```
x_ts = np.c_[x_lon, x_lat, x_alt, x_spe, x_head]
y_ts = np.c_[y_lon, y_lat, y_alt]
```

Αλγόριθμος Νευρωνικών Δικτύων

```
model = Sequential()
model.add(LSTM(32, input_shape=(look_back, x.shape[2])))
model.add(Dense(16, activation='relu'))
model.add(Dense(3, activation='sigmoid'))
model.compile(loss='mean_squared_error', optimizer='adam')
time_callback = TimeHistory()
history = model.fit(x, y, batch_size=512, epochs=50, validation_data=(x_ts, y_ts), callbacks=[time_callback], verbose=1)
times = time_callback.times
sum(times)
```

Δημιουργία πίνακα για Multi Step Prediction

```
predictions = []
new_x = x_ts[:1] # πρώτη χρονική στιγμή
for _ in tqdm(range(1, len(y_ts))):
preds = model.predict(new_x) # προβλέπουμε την επόμενη χρονική στιγμή
new_x = np.r_[new_x[0, 1:, :], preds].reshape(1, 5, 5)  # το ενώνουμε με τις 4 τελευταίες τιμές του x
predictions.append(preds)

predictions = np.stack(predictions).reshape(-1, 5)
```

Βιβλιογραφία

Ayhan, S., & Samet, H. (2016). *Aircraft Trajectory Prediction Made Easy with Predictive Analytics.* San Francisco, CA, USA.

Bacao, F., Lobo, V., & Painho, M. (2005). *On the particular characteristics of spatial data and its similarities to secondary data used in data mining.*

Beam, A. L. (2017). Deep Learning 101 - Part 1: History and Background. Στο *machine learning and medicine* .

Brownlee, J. (2016, July 21). *machinelearningmastery*. Ανάκτηση από https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

Brownlee, J. (2017). A Gentle Introduction to Backpropagation Through Time. *Machine Learning Mastery*.

Buchin, M., Driemel, A., Kreveld, M. v., & Sacristan, V. (2011). Segmenting trajectories: A framework and algorithms using spatiotemporal criteria. *Journal of Spatial Information Science*.

Copeland, M. (2016). What's the Difference Between Artificial Intelligence, Machine Learning, and Deep Learning?

Deoras, S. (2017). How Is Machine Learning Different From Statistics. *Analytics India*.

Foote, K. D. (2017). A Brief History of Deep Learning. *Dataversity*.

Georgiou, H., Karagiorgou, S., Kontoulis, Y., Pelekis, N., Petrou, P., Scarlatti, D., & Theodoridis, Y. (2018). *Predicting the Next Steps of Moving Objects: A Survey* .

Gers, F. (2001). *Long Short-Term Memory in Recurrent Neural Networks.* ECole Polytechnique FeDeRale De Lausanne.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.*

Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks.* Springer-Verlag Berlin Heidelberg.

Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. (2017). *LSTM: A Search Space Odyssey.* arXiv: 1503.04069v2 [cs.NE].

Guo, J. (2013). *BackPropagation Through Time.*

Jaeger, H. (2013). *A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach* .

Karatzoglou, A., Jablonski, A., & Beigl, M. (2018). *A Seq2seq Learning Approach for Modeling Semantic Trajectories and Predicting the Next Location.*

Lu, Y., & Salem, F. M. (χ.χ.). *Simplified Gating in Long Short-term Memory (LSTM) Recurrent Neural Networks, .* Michigan State University, USA.

Nielsen, M. (2015). *Neural Network and Deep Learning.* Determination Press.

Olah, C. (2015). *Understanding LSTM Networks.*

Pelekis, N., & Theodoridis, Y. (2014). *Mobility Data Management and Exploration.* Springer New York Heidelberg Dordrecht London.

Saghafian, B. (2013). *Interpolation.*

Schmidhuber, J., & Hochreiter, S. (1997). *Long Short Term Memory.* Neural Computation 9(8):1735-1780.

Spinsanti, L., Celli, F., & Renso, C. (2010). *Where you stop is who you are:understanding people's activities by places visited.* 5th BMI'10, User Behaviour Modelling, Karlsruhe, Germany.

Stansbury, D. (2014). Derivation: Derivatives for Common Neural Network Activation Functions. *The Clever Machine*.

Thomas, A. (2018). A brief introduction to LSTM networks. *Adventures in machine learning*.

Werbos, P. (1990). *Backpropagation through time: what it does and how to do it.* Proceedings of the IEEE 78.

wikipedia. (χ.χ.).

Woodie, A. (2017). Why Deep Learning, and Why Now. *datanami*.

Yan, Z. (2010). *Traj-ARIMA: A Spatial-Time Series Model for Network-Constrained Trajectory.*

Yao, D., Zhang, C., Huang, J., & Bi, J. (2017). *SERM: A Recurrent Model for Next Location Prediction in Semantic Trajectories.* Singapore.

Yu, Li, Dong, & Deng. (2014). *Deep Learning - Methods and Applications.*

Μήτσιος, Γ. (2017). *Αναδρομικά Νευρωνικά Δίκτυα και αυτόματη παραγωγή Hashtag από Tweet του Twitter.*

Πλεύρου, Α. (2012). *Τεχνητά νευρωνικά δίκτυα προσομοίωσης του ανθρωπίνου εγκεφάλου.*

Πνευματικός, Σ. (2006). Η κλασική θεώρηση του χώρου και του χρόνου. Στο *Κλασική Μηχανική.*

Ρεφανίδης, Γ. (2011). *Νευρωνικά Δίκτυα.*

Τραγοπούλου, Σ. (2013). *Συλλογή και κατηγοριοποίηση Δεδομένων Κίνηση.* Χαροκόπειο Πανεπιστήμιο.

Τριανταφυλιδου, Δ. (2016). *Ανιχνευση πορσωπων με βαθια συνελεκτικα δικτυα.*

Χονδροδήμα, Ε. (2013). *Πρόβλεψη σε ημερολόγια Κίνησης με χρήση Νευρωνικών Δικτύων.*