ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

ΤΜΗΜΑ

ΣΤΑΤΙΣΤΙΚΗΣ ΚΑΙ ΑΝΑΛΟΓΙΣΤΙΚΩΝ-ΧΡΗΜΑΤΟΟΙΚΟΝΟΜΙΚΩΝ ΜΑΘΗΜΑΤΙΚΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Machine learning και αναλογισμός

Συγγραφέας:
Ανδρόνικος Δήμας

Επιβλέπων:
Πέτρος Χατζόπουλος

21 Ιουνίου 2018

Τριμελής Επιτροπή

Χατζόπουλος Πέτρος, Επίκουρος Καθηγητής

Καραγρηγορίου Αλέξανδρος, Καθηγητής

Μαραγκουδάκης Εμμανουήλ, Αναπληρωτής Καθηγητής

Ευχαριστίες

Ευχαριστώ πολύ τον επιβλέποντα καθηγητή μου, κύριο Χατζόπουλο Πέτρο, για την βοήθεια και την υποστήριξη που μου προσέφερε κατά την διάρκεια της εκπόνησης αυτής της εργασίας.

Ευχαριστώ τους καθηγητές κύριο Αλέξανδρο Καραγρηγορίου και κύριο Μανώλη Μαραγκουδάκη για τη συμμετοχή τους στην επιτροπή.

Θέλω να ευχαριστήσω ιδιαίτερα τον κύριο Αλέξανδρο Καραγρηγορίου, για την διδασκαλία του μέσα στα χρόνια των σπουδών μου, την καθοδήγηση του στα ακαδημαϊκά ζητήματα και την βοήθεια του στην συγγραφή αυτής της εργασίας.

Ευχαριστώ επίσης τους συμφοιτητές μου Α. Αναστασίου, Κ. Λαδόπουλο και Π. Τρέντου για τη συμβολή τους στην προσπάθεια μου, αυτή.

Τέλος, ευχαριστώ θέρμα την οικογένειά μου για την υποστήριξή τους καθόλη την διάρκεια των σπουδών μου.

Περιεχόμενα

ΕΙΣΑΓΩΓΗ

ΚΕΦΑΛΑΙΟ 1: ΜΕΘΟΔΟΙ ΚΑΙ ΤΕΧΝΙΚΕΣ MACHINE LEARNING

ΚΕΦΑΛΑΙΟ 2 ΜΟΝΤΕΛΑ ΘΝΗΣΙΜΟΤΗΤΑΣ

ΚΕΦΑΛΑΙΟ 3 ΜΙΑ ΕΦΑΡΜΟΓΗ ΣΤΟΝ ΑΝΑΛΟΓΙΣΜΟ

Περίληψη

Στόχος αυτής της πτυχιακής εργασίας είναι η εφαρμογή ενός machine learning αλγορίθμου για την επίλυση προβλημάτων στον κλάδο των αναλογιστικών μαθηματικών. Πιο συγκεκριμένα το πρόβλημα το οποίο προσπαθούμε να λύσουμε σε αυτήν την εργασία είναι η πρόβλεψη χρονοσειρών. Ο τρόπος με τον οποίο θα λύσουμε αυτό το πρόβλημα είναι χρησιμοποιώντας ένα artificial neural network (τεχνητό νευρωνικό δίκτυο). Ειδικότερα θα χρησιμοποιήσουμε long-short term memory μονάδες οι οποίοι χρησιμοποιούνται με μεγάλη επιτυχία σε δεδομένα ακολουθιών, όπως ο ήχος, η γραφή, οι χρονοσειρές. Σε αυτή την εργασία αρχικά, θα δοθεί η απαραίτητη θεωρία ούτως ώστε να μπορέσει να κατανοηθεί η φιλοσοφία ενός long-short term memory και ύστερα θα το εφαρμόσουμε πάνω σε μια χρονοσειρά αναλογιστικών δεδομένων προκειμένου να εξετάσουμε τον βαθμό στον οποίο επιτυγχάνει την πρόβλεψη σε τέτοιας μορφής δεδομένα.

ΕΙΣΑΓΩΓΗ

0.1. Τι είναι το machine learning ?

Με την έννοια machine learning αναφερόμαστε στον προγραμματισμό αλγόριθμων οι οποίοι έχουν την ικανότητα να μαθαίνουν από την εμπειρία τους με τα δεδομένα. Είναι ένα υποπεδίο της τεχνητής νοημοσύνης και οι αλγόριθμοι αυτοί χρησιμοποιούνται για την κατηγοριοποίηση δεδομένων, την ομαδοποίηση δεδομένων, παλινδρόμηση, αναγνώριση πρότυπου, αναγνώριση ανωμαλιών και την πρόβλεψη.

Το machine learning είναι ουσιαστικά μία μορφή εφαρμοσμένης στατιστικής με έμφαση στην χρήση υπολογιστών και μεταξύ άλλων την εκτίμηση παραμέτρων πολύπλοκων συναρτήσεων.

Ως τεχνητή νοημοσύνη ορίζουμε την ικανότητα μιας μηχανής να επεξεργάζεται το περιβάλλον της και να πράττει έτσι ώστε να μεγιστοποιεί τις πιθανότητες επιτυχίας του εκάστοτε στόχου της. Για λεπτομέρειες ο αναγνώστης μπορεί να αποταθεί στους Mitchell (1997), Mohri (2012), Samuel (1959; 1988).

0.2 Διάφοροι τρόποι μάθησης

Οι machine learning αλγόριθμοι μπορούν να κατηγοριοποιηθούν γενικά σε τρεις κατηγορίες ανάλογα με το τι διαδικασία μάθησης που χρησιμοποιούν. Αυτές οι κατηγορίες είναι οι εξής :

1) Unsupervised learning

Οι Unsupervised learning αλγόριθμοι προσπαθούν να μάθουν τη δομή των δεδομένων και τις ιδιότητες των χαρακτηριστικών τους. Συνήθως θέλουμε να μάθουμε την κατανομή πιθανότητας που παρήγαγε αυτό το σύνολο δεδομένων. Με στατιστικούς όρους θέλουμε να εκτιμήσουμε την πυκνότητα της κατανομής.

2) Supervised learning

Οι Supervised learning αλγόριθμοι προσπαθούν να βρούνε τον κανόνα με τον οποίο συνδέονται το input (εξαρτημένες μεταβλητές) με το output (ανεξάρτητη μεταβλητή) όπου εμείς οι

επιβλέποντες (supervisors) παρέχουμε τις σωστές τιμές που θα έπρεπε να παίρνει το output. Ο τελικός στόχος είναι να προβλέψουμε το outputέχοντας ένα input.

Βέβαια υπάρχει και το semi-supervised learning όπου χρησιμοποιούνται και οι δύο μέθοδοι, δηλαδή σε κάποια παραδείγματα κατά την διάρκεια της μάθησης δίνεται η σωστή απάντηση και σε κάποια όχι.

3) Reinforcement learning

Οι Reinforcement learning αλγόριθμοι προσπαθούν να διαλέξουν πράξεις ή μία ακολουθία πράξεων με σκοπό να μεγιστοποιήσουν την ‘ανταμοιβή’.

0.3. Training set και test set

Ο γενικός στόχος μας είναι ο αλγόριθμος να λειτουργεί καλά και σε καινούργια δεδομένα τα οποία δεν έχει χρησιμοποιήσει για την εκπαίδευσή του.

Όταν εκπαιδεύουμε ένα machine learning μοντέλο χρησιμοποιούμε ένα set δεδομένων για να εκτιμήσουμε τις παραμέτρους του. Αυτό το set δεδομένων το ονομάζουμε training set. Ύστερα για να ελέγξουμε πόσο καλό είναι αυτό το μοντέλο παίρνουμε ένα άλλο set δεδομένων το οποίο το ονομάζουμε test set το οποίο είναι διάφορο του training set. Ο σκοπός μας είναι και στα δύο αυτά set το σφάλμα του μοντέλου μας να είναι μικρό. Η αποτελεσματικότητα του αλγορίθμου και του προτεινόμενου μοντέλου καθορίζεται σε μεγάλο βαθμό από το μέγεθος του σφάλματος του test set.

Επιπλέον κάνουμε και κάποιες υποθέσεις για αυτά τα set δεδομένων οι οποίες είναι :

(α) Τα δύο αυτά set δεδομένων πρέπει να προέρχονται από την ίδια κατανομή και

(β) ότι τα παρατηρήσεις (παραδείγματα) σε κάθε set δεδομένων είναι ανεξάρτητες μεταξύ τους.

Από εδώ προκύπτουν και οι δύο βασικοί παράγοντες που καθορίζουν το πόσο καλά θα λειτουργήσει ο αλγόριθμος.

Ο ένας είναι το training error να είναι μικρό και ο άλλος το test error να μην απέχει πολύ από το training error.

0.4. Deep Learning

Η πραγματική δυσκολία που αντιμετωπίζει η τεχνητή νοημοσύνη είναι να κάνει λειτουργίες που είναι εύκολες για τον άνθρωπο αλλά είναι δύσκολο να περιγράψει ο ίδιος πως τις κάνει. Για παράδειγμα την αναγνώριση εικόνων, ήχων κτλ. Ο κλάδος που ασχολείται με αυτό και λύνει αυτό το πρόβλημα είναι το Deep Learning. Το Deep Learning είναι ένα υποπεδίο του machine learning. Η διαδικασία που ακολουθεί το Deep Learning είναι να μαθαίνει από την εμπειρία και να καταλαβαίνει τον κόσμο μέσα από μία διαδικασία ιεραρχικών εννοιών, με κάθε έννοια να καθορίζεται εν μέσω της σχέσης της με μία απλούστερη έννοια. Η ιεραρχία των εννοιών επιτρέπει στον υπολογιστή να μαθαίνει σύνθετες έννοιες μέσω πιο απλών εννοιών. Αν δείξουμε διαγραμματικά αυτήν την μέθοδο, θα δούμε ότι αυτές οι έννοιες είναι χτισμένες η μία πάνω στην άλλη και το διάγραμμα είναι ‘’βαθύ (deep)’’ με πολλά στρώματα (layers). Από εκεί λοιπόν προκύπτει και το όνομα Deep Learning.

Το Deep Learning δεν είναι κάτι καινούργιο. Αυτό το πεδίο έχει ξεκινήσει από το 1940 και έχει κατά καιρούς αλλάξει πολλά ονόματα. Το πρώτο ρεύμα μελέτης ονομαζόταν Cybernetics και ξεκίνησε την περίοδο 1940 – 1960, το δεύτερο ρεύμα ονομάστηκε Connectionism και ξεκίνησε την περίοδο 1980 – 1990 και εν τέλει ονομάστηκε Deep Learning από το 2006.

Μερικοί από τους πρώτους αλγόριθμους μάθησης φτιάχτηκαν με τον σκοπό να προσομοιώσουν τον τρόπο με τον οποίο γίνεται η μάθηση στον εγκέφαλο. Για αυτό τον λόγο το Deep learning ονομάζεται και artificial neural networks (ANNs) (Τεχνητά νευρωνικά δίκτυα). Δεν είναι ρεαλιστικά μοντέλα για το πως ένα βιολογικό νευρωνικό δίκτυο λειτουργεί αλλά ακολουθούν τις ίδιες υπολογιστικές αρχές.

Με την πρόοδο της τεχνολογίας τις τελευταίες δεκαετίες αναδείχθηκε πολύ η σπουδαιότητά αυτών των αλγορίθμων διότι η υπολογιστική δύναμη των υπολογιστών μεγάλωσε φοβερά, ο όγκος των διαθέσιμων δεδομένων αυξήθηκε και συνεχίζει να αυξάνεται εκθετικά και οι μέθοδοι εκπαίδευσης των νευρωνικών δικτύων βελτιώθηκαν κάνοντας έτσι αυτά τα μοντέλα πάρα πολύ αποτελεσματικά.

ΚΕΦΑΛΑΙΟ 1

ΜΕΘΟΔΟΙ ΚΑΙ ΤΕΧΝΙΚΕΣ MACHINE LEARNING

1.1. Gradient descend

Οι περισσότεροι αλγόριθμοι machine learning περιλαμβάνουν και κάποια βελτιστοποίηση. Λέγοντας βελτιστοποίηση εννοούμε την διαδικασία μεγιστοποίησης ή ελαχιστοποίησης μια συνάρτησης $f(x)$ ως προς x (Το x μπορεί να δηλώνει και διάνυσμα).

Η συνάρτηση που θέλουμε να ελαχιστοποιήσουμε ή να μεγιστοποιήσουμε καλείται objective function ή criterion. Στην περίπτωση της ελαχιστοποίησης χρησιμοποιούνται και οι όροι loss function, cost function ή και error function.

Συνήθως στο machine learning οι συναρτήσεις που έχουμε να ελαχιστοποιήσουμε είναι πολυμεταβλητές και η εύρεση του ολικού ελάχιστου είναι μια πολύ δύσκολη έως και αδύνατη διαδικασία. Έτσι αρκούμαστε στο να βρίσκουμε ένα τοπικό ελάχιστο ή και ακόμα ένα σημείο το οποίο δεν είναι ούτε ολικό ούτε τοπικό ελάχιστο αλλά μικραίνει αρκετά την cost function. Η διαδικασία με την οποία γίνεται αυτό περιγράφεται παρακάτω.

Η έννοια του gradient γενικεύει την έννοια της παραγώγου και είναι το διάνυσμα όλων των μερικών παραγώγων της πολυμεταβλητής συνάρτησης f:$R^n \to R, f(x_1, x_2, \dots, x_n)$ = f (x), $x = (x_1, x_2, \dots, x_n)'$ και συμβολίζεται ως

$$\nabla_x f(x) = \begin{pmatrix} \frac{d}{dx_1} f(x) \\ \frac{d}{dx_2} f(x) \\ \vdots \\ \frac{d}{dx_n} f(x) \end{pmatrix}$$

Γνωρίζουμε από τον απειροστικό λογισμό ότι το gradient μας υποδεικνύει την κατεύθυνση όπου μεγιστοποιείται η συνάρτηση. Επομένως προκειμένου να μειώσουμε την f κινούμαστε στην

αρνητική κατεύθυνση από αυτή που μας υποδεικνύει το gradient. Αυτή η μέθοδος είναι γνωστή ως method of steepest descend ή gradient descend.

Αυτή η μέθοδος προτείνει ένα καινούργιο σημείο το :

$$x' = x - \varepsilon \nabla_x f(x)$$

όπου το ε είναι ο ρυθμός μάθησης, είναι βαθμωτό και καθορίζει το βήμα με το οποίο κινούμαστε προς το ελάχιστο. Με αυτό τον τρόπο κατεβαίνουμε σιγά σιγά προς τα χαμηλότερα σημεία της συνάρτησης μέχρι να βρούμε ένα ικανοποιητικό χαμηλό σημείο (εκεί όπου θα μηδενίζεται η παράγωγος συνήθως εκεί όπου η συνάρτηση κόστους θα σταματήσει να μικραίνει).

Εφαρμόζοντας αυτήν την μέθοδο σε ένα training set με n αριθμό παρατηρήσεων (παραδειγμάτων), έστω $x_1, x_2, \dots, x_n$,για να υπολογίσουμε το gradient,θα υπολογίσουμε ξεχωριστά το gradient για κάθε παράδειγμα και μετά θα υπολογίσουμε τον μέσο όρο αυτών.

$$\nabla f = \frac{1}{n} \sum_{i=1}^{n} \nabla f(x_i)$$

1.2. Stochastic gradient descend

Το πρόβλημα με τη μέθοδο gradient descent είναι ότι είναι υπολογιστικά πολύ δαπανηρή και καθώς μπορεί να είναι πολύ μεγάλο το training set και να απαιτείται πολύς χρόνος για να υπολογιστεί κάθε βήμα προς την κατάβαση. Μία λύση σε αυτό είναι η μέθοδος stochastic gradient descend όπου αντί να υπολογίζουμε το gradient με βάση ολόκληρο το training set, διαλέγουμε ένα τυχαίο δείγμα από το training set και υπολογίζουμε το gradient και με αυτόν τον τρόπο εκτιμούμε σημειακά το gradient όλου του training set. Για κάθε βήμα κατάβασης ως προς το επιθυμητό σημείο χρησιμοποιούμε διαφορετικό τυχαίο δείγμα και επανεκτιμούμε το gradient βάση αυτού. Ο πληθυκός αριθμός του υποσυνόλου αυτού πρέπει να είναι μικρός με εύρος από ένα έως μερικές εκατοντάδες.

Έστω δηλαδή ότι διαλέγουμε ένα μικρό δείγμα m παρατηρήσεων από τις n, $x_1, x_2, \dots, x_m$, τότε ο εκτιμητής της ∇f θα είναι :

$$\nabla \hat{f} = \frac{1}{m} \sum_{i=1}^{m} \nabla f(x_i)$$

Κατά την διαδικασία της εκπαίδευσης ενός τεχνητού νευρωνικού δικτύου με την μέθοδο stochastic gradient descent κάθε φορά που τα δείγματα τα οποία χρησιμοποιούμε για την εκτίμηση

έχουν καλύψει πλήρως τον δειγματικό μας χώρο, δηλαδή για την εκτίμηση του gradient έχουν χρησιμοποιηθεί όλες οι διαθέσιμες παρατηρήσεις λέμε ότι έχει συμπληρωθεί ένα epoch. Στην πράξη όταν εκπαιδεύουμε ένα τεχνητό νευρωνικό δίκτυο μπορεί να χρειαστούν πολλά epochs.Το ίδιο ισχύει και για την μέθοδο gradient descend όταν έχουμε καλύψει όλον τον δειγματικό χώρο κατά την διάρκεια της εκπαίδευσης και σπεύδουμε να επαναλάβουμε την διαδικασία.

1.3. Perceptron

Ένα από τα πιο απλά είδη τεχνητού νευρώνα ονομάζεται perceptron. Ο τρόπος που ένας perceptron λειτουργεί είναι ο εξής: Ο perceptron είναι μία συνάρτηση παίρνει διάφορες τιμές από κάποια inputs, π.χ. $x_1, x_2, \dots, x_n$τα όποια είναι δίτιμες τυχαίες μεταβλητές (παίρνουν τιμές στο 0 ή 1), και μας δίνει πίσω πάλι ένα δίτιμο output (y). Δεν έχουν όλα τα inputs την ίδια αξία. Ο τρόπος με τον οποίο μπορούμε να το μοντελοποιήσουμε αυτό είναι βάζοντας βάρη στα inputs, $w_1, w_2, \dots, w_n$ ανάλογα με την σημασία τους στον υπολογισμό του output. Έτσι η τιμή που θα πάρει ο νευρώνας y (0 ή 1) καθορίζεται από τον αν το σταθμισμένο άθροισμα $\sum_{i=1}^{n} w_i x_i$, είναι μεγαλύτερο ή μικρότερο από ένα αριθμητικό όριο (threshold value). Τα βάρη και το αριθμητικό όριο είναι πραγματικοί αριθμοί.

$$y = \begin{cases} 0, & \alpha\nu \sum_{i=1}^{n} w_i x_i \leq threshold \\ 1, & \alpha\nu \sum_{i=1}^{n} w_i x_i > threshold \end{cases}$$

Ένας καλύτερος τρόπος να γραφεί το παραπάνω είναι να συμβολίσουμε με – b το αριθμητικό όριο, το οποίο το ονομάζουμε μεροληψία (bias). Δηλαδή, $threshold\ value = -b$. Έτσι η παραπάνω συνάρτηση θα μπορεί να γραφεί :

$$y = \begin{cases} 0, & \alpha\nu \sum_{i=1}^{n} w_i x_i + b \leq 0 \\ 1, & \alpha\nu \sum_{i=1}^{n} w_i x_i + b > 0 \end{cases}$$

Αν θέλουμε να το γράψουμε και με όρους γραμμικής άλγεβρας, όπου w είναι το διάνυσμα των βαρών, $w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix}$ και x είναι το διάνυσμα των inputs, $x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$, τότε η παραπάνω εξίσωση γράφεται :

$$y = \begin{cases} 0, & \alpha v \ w^T x + b \ \leq 0 \\ 1, & \alpha v \ w^T \ x + b \ > 0 \end{cases}$$

όπου w^Tσυμβολίζει το ανάστροφο διάνυσμα του w. Το bias είναι ένα μέτρο του κατά πόσο εύκολα ένα perceptron μας δίνει σαν output την τιμή 1.

1.4. Sigmoid Neurons and artificial neurons

Οι sigmoid νευρώνες έχουν την ίδια δομή με τους perceptron με τη μόνη διαφορά ότι τα inputs $x_1, x_2, \ldots \ , x_n$ μπορούν να πάρουν οποιαδήποτε τιμή ανάμεσα στο 0 και το 1. Για κάθε input x_i αντιστοιχεί πάλι ένα βάρος w_i , i = 1,...,n. Και υπάρχει και εδώ η μεροληψία b.Το output εδώ δεν παίρνει την τιμή 0 ή 1 αλλά την τιμή που δίνει η συνάρτηση $y = \sigma(w^T x + b)$, όπου σ είναι η sigmoid function ή αλλιώς logistic function και μας δίνει output ανάμεσα στο 0 και το 1 και ορίζεται ως :

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

Άρα, στην περίπτωσή μας,

$$z = \ w^T \ x + b$$

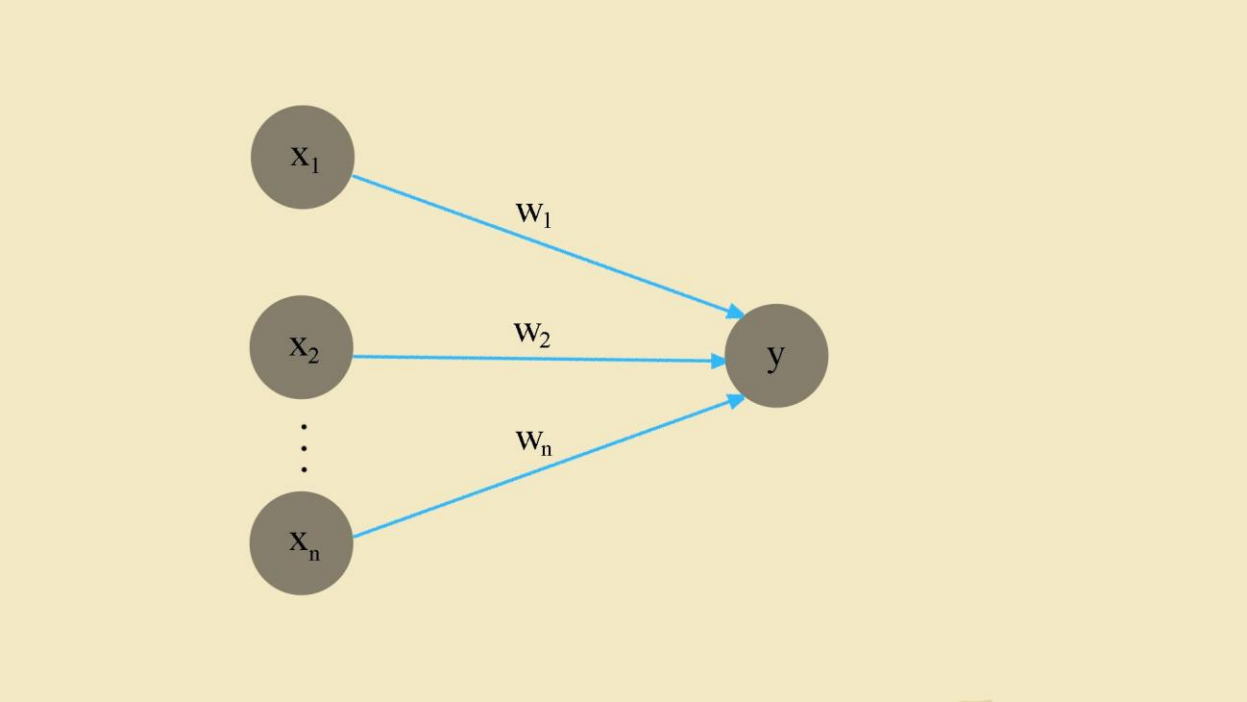
$$y = \sigma(w^T x + b) = \frac{1}{1 + \ e^{-(w^T x + b)}}$$

ή $y = \sigma(z) = \frac{1}{1 + e^{-z}}$

Η συνάρτηση η οποία χρησιμοποιείται για να υπολογιστεί το output του τεχνητού νευρώνα καλείται activation function. Σε αυτήν την περίπτωση είναι η logistic αλλά μπορούμε γενικά να χρησιμοποιήσουμε και πολλές άλλες συναρτήσεις αντί αυτής αναλόγως της χρήσης για τη χρήση προορίζουμε τον εκάστοτε νευρώνα.

Έτσι μπορούμε να δώσουμε ένα πιο γενικό ορισμό σε έναν τεχνητό νευρώνα ως μία συνάρτηση η οποία υπολογίζει ένα σταθμισμένο άθροισμα y με βάση ένα διάνυσμα μεταβλητών χ και με βάρη το διάνυσμα w, σε αυτό προσθέτει ένα συγκεκριμένο αριθμό που λέγεται bias(b) και στο τέλος εφαρμόζει σε αυτό το σταθμισμένο άθροισμα μία συνάρτηση g την οποία καλούμε activation function. Δηλαδή (βλέπε εικόνα 1.1),

$$y = g(w^T x + b)$$



Εικόνα 1.1: Τεχνητός Νευρώνας (artificial neuron).

Οι τρόποι που περιγράψαμε παραπάνω, gradient descend και stochastic gradient descend χρησιμοποιούνται για την εκτίμηση των weights και των biases. Αφού πρώτα βάλουμε κάποιες αρχικές τυχαίες τιμές σε αυτές τις παραμέτρους. Ύστερα χρησιμοποιώντας μία συνάρτηση που ονομάζουμε cost function συγκρίνουμε την τιμή που εκτίμησε το τεχνητό νευρωνικό δίκτυο με αυτήν που θα θέλαμε να είχε παράξει και μετά με τις gradient descend ή stochastic gradient descent ή και άλλες μεθόδους αλλάζουμε τις τιμές των παραμέτρων biases και weights έτσι ώστε να μικρύνουμε κατά το όσο είναι δυνατόν την τιμή της cost function.

1.5. Δομή των Neural networks

Τώρα, αν συνδυάσουμε πολλούς artificial neurons όπως την εικόνα 1.2, αυτό που φτιάχνουμε είναι ένα τεχνητό νευρωνικό δίκτυο (artificial neural network). Σαν layer ορίζουμε ένα σύνολο νευρώνων στο οποίο γίνονται παράλληλοι υπολογισμοί. Το πρώτο layer από το οποίο ξεκινάει η πληροφορία ονομάζεται input layer και οι νευρώνες που ανήκουν σε αυτό input neurons. Το τελευταίο layer από το οποίο βγαίνει το συνολικό αποτέλεσμα της λειτουργίας του νευρωνικού δικτύου ονομάζεται output layer και οι νευρώνες που ανήκουν σε αυτό output neurons. Ότι ενδιάμεσο στρώμα υπάρχει ονομάζεται hidden layer. Αξίζει να σημειωθεί ότι ενώ ο σχεδιασμός του input layer και του output layer είναι ξεκάθαρος συνήθως από το πρόβλημα για το οποίο θέλουμε το τεχνητό νευρωνικό δίκτυο να μας λύσει, ο σχεδιασμός των hidden layers είναι μια αυθαίρετη διαδικασία η οποία δεν ακολουθεί κάποιον κανόνα αλλά την διαίσθηση του δημιουργού και την εμπειρία και γνώση από προηγούμενες αρχιτεκτονικές δικτύων.

Τα νευρωνικά δίκτυα όπου κάθε layer χρησιμοποιεί σαν input το output του προηγούμενου layerονομάζονται feed forward γιατί η πληροφορία κυλάει πάντα προς τα μπροστά, δηλαδή ένα layer μπορεί να επικοινωνεί μόνο με ένα άλλο layer και με κατεύθυνση προς το output layer. Υπάρχουν και τεχνητά νευρωνικά δίκτια που επιτρέπουν την διάδοση πληροφορίας μεταξύ των hidden layers μέσα στον χρόνο. Αυτά ονομάζονται recurrent neural networks. Θα αναφερθούμε ειδικότερα σε αυτά παρακάτω.

1.6. Deep Feed Forward network

Τα deep forward networks ή feed forward neural networks ή ακόμα και multilayer perceptrons είναι βασικά για την κατανόηση οποιουδήποτε πιο σύνθετου μοντέλου τεχνητών νευρωνικών δικτύων. Ο στόχος ενός τέτοιου νευρωνικού δικτύου είναι να προσεγγίσει κάποια συνάρτηση f. Για παράδειγμα ένα νευρωνικό δίκτυο που θέλουμε να κάνει την δουλειά της κατηγοριοποίησης, $y = f(x)$ όπου αντιστοιχεί το χ που είναι το input (ανεξάρτητη μεταβλητή) και το y, το οποίο είναι το output (εξαρτημένη μεταβλητή) και είναι μια κατηγορία. Ένα τέτοιο νευρωνικό δίκτυο ορίζει με την συνάρτησή του $y = f(x, \theta)$ τον κανόνα με τον οποίο συνδέονται οι δύο αυτές μεταβλητές και μαθαίνει/εκτιμά το διάνυσμα των παραμέτρων θ που θα οδηγήσει στην καλύτερη απόδοση του μοντέλου.

Αυτά τα μοντέλα καλούνται feedforward διότι η πληροφορία που εισάγεται από το διάνυσμα x του input κινείται προς τα μία μόνο κατεύθυνση μέχρι να μας δώσει το output. Αυτό είναι πιο εύκολο να γίνει αντιληπτό διαγραμματικά. Ένας εύκολος τρόπος να αντιληφθούμε τα deep feedforward networks είναι να συνδυάσουμε την προηγούμενη γνώση μας για τους artificial neurons. Έστω ότι έχουμε ένα σύνολο από artificial neurons και αυτό το σύνολο το στοιχίζουμε σε μία γραμμή. Τώρα, έστω ότι έχουμε ξανά ένα σύνολο από artificial neurons και το στοιχίζουμε

και αυτό σε μια γραμμή. Συνεχίζουμε με αυτόν τον τρόπο να δημιουργούμε θεωρητικά όσες γραμμές θέλουμε. Τώρα αυτές οι γραμμές από νευρώνες αντιπροσωπεύουν τα layers. Η πρώτη που δημιουργήσαμε είναι το input layer, η τελευταία το output layer και όλες οι ενδιάμεσες τα hidden layers. Τέλος, μπροστά θα είναι η κατεύθυνση από την γραμμή του input layer προς την γραμμή του output layer. Ο κανόνας που κάνει ένα artificial neural network να είναι forward είναι ότι, σε κάθε layer οι νευρώνες μεταδίδουν πληροφορία μόνο στο αμέσως μπροστά τους layer.

Ένα παράδειγμα είναι στην εικόνα 1.2, όπου η πρώτη στήλη με τα x1, x2, x3, x4, x5, x6 είναι τα inputs και τρόπος που γίνεται ο υπολογισμός είναι με κατεύθυνση προς τα δεξιά. Το δίκτυο αυτό αποτελείται από 4 layers (πιο κάτω ορίζεται τι είναι το layer).

Ονομάζονται networksδιότι αποτελούνται από την σύνθεση πολλών συναρτήσεων μαζί. Για παράδειγμα μπορεί να έχουμε 3 συναρτήσεις τις f^2, f^3 $και$ f^4(οι αριθμοί πάνω δεξια στις συναρτήσεις δεν συμβολίζουν δυνάμεις, αλλά είναι απλοί δείκτες που δηλώνουν την συγκεκριμένη συνάρτηση) οι οποίες θα είναι συνδεδεμένες με τον εξής τρόπο $f^2(\ f^3(f^4(x)))$. Η f^2θα λέγεται second layer, η f^3 θα λέγεται third layer και με παρόμοια λογική ονομάζονται και τα επόμενα. Η τελευταία συνάρτηση καλείται output layer. Ο πληθυκός αριθμός των συναρτήσεων μας δίνει το depth (βάθος) του μοντέλου. Τα layers ενδιάμεσα στο first layer και στο output layer καλούνται hidden layers.Το first layer μπορούμε να το σκεφτούμε όχι σαν συνάρτηση αλλά απλώς το σύνολο των μεταβλητών inputs. Κατά την διαδικασία της μάθησης, ο machine learning αλγόριθμος προσπαθεί να μάθει πως να χρησιμοποιεί αυτά τα layers για να παράγει το επιθυμητό output. Κάθε layer περιέχει μονάδες ή κόμβους ή νευρώνες τα οποία είναι μεταβλητές οι οποίες παίρνουν κάποια αριθμητική τιμή. Δεν υπάρχει κάποιος κανόνας που να καθορίζει τον αριθμό των layers και των νευρώνων σε κάθε layer. Φυσικά ανάλογα με την χρήση για την οποία δημιουργείται το κάθε νευρωνικό δίκτυο ο αριθμός των νευρώνων στο input layer και στο output layer μπορεί να είναι προφανής. Για παράδειγμα, αν χρησιμοποιούσαμε ένα deep forward neural network για να κατηγοριοποιήσουμε εικόνες ο αριθμός των input neurons θα ήταν όσο και τα pixel της εικόνας ενώ ο αριθμός των output layer neurons θα ήταν όσες κατηγορίες θα θέλαμε το δίκτυο να έχει σαν επιλογή να κατηγοριοποιήσει.

Έτσι θα έχουμε,

$$f^{(2)} = g^{(2)}(W^{(2)}x + b^{(2)})$$

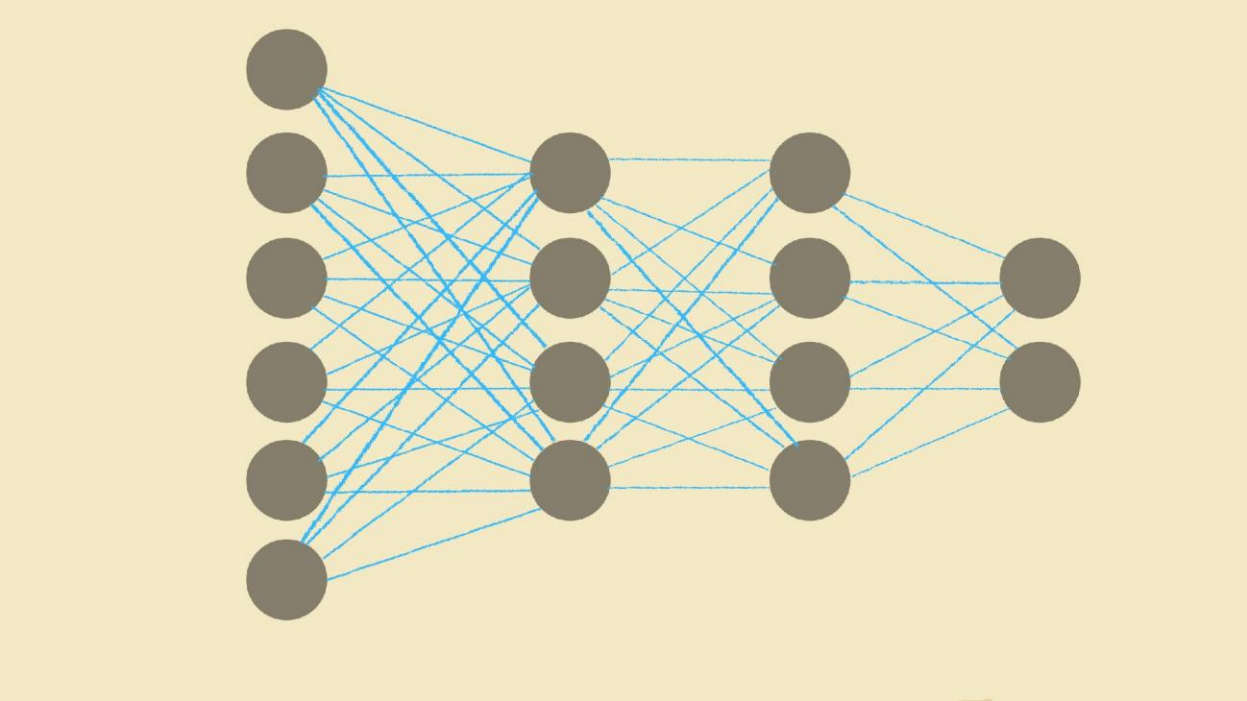
Όπου$f^{(2)}$το first layer, $g^{(2)}$η activation function του layer, $W^{(2)}$ο πίνακας των βαρών, x το διάνυσμα των inputs και $b^{(2)}$ το διάνυσμα των biases.

Με παρόμοιο τρόπο υπολογίζουμε και το second layer,

$$f^{(3)} = g^{(3)}(W^{(3)}f^{(2)} + b^{(3)})$$

Συνεχίζοντας ακριβώς με τον ίδιο τρόπο και τα επόμενα.

Ένα εικονογραφημένο παράδειγμα είναι το παρακάτω εικόνα 1.2:



Εικόνα 1.2: Απλό νευρωνικό δίκτυο (artificial neural network, ANN).

Τέλος αυτά τα δίκτυα ονομάζονται νευρωνικά διότι η έμπνευση γι' αυτού του είδους αλγορίθμους έρχεται από τα βιολογικά νευρωνικά δίκτυα.

Ένας βιολογικός νευρώνας έχει την εξής δομή. Αποτελείται από το κυτταρικό του σώμα, έναν άξονα (axon) με τον οποίο στέλνει πληροφορία σε άλλους νευρώνες και δενδρίτες από τους οποίους παίρνει πληροφορία από άλλους νευρώνες. Το σημείο όπου ο άξονας ενός νευρώνα έρχεται σε επαφή με τον δενδρίτη ενός άλλου νευρώνα ονομάζεται νευρική σύναψη. Έχει ανακαλυφθεί ότι ο εγκέφαλος μαθαίνει αλλάζοντας την ισχύ της νευρικής σύναψης κατά την επαναλαμβανόμενη διέγερση από το ίδιο ερέθισμα.

Με παρόμοιο τρόπο ένα τεχνητό νευρωνικό δίκτυο αποτελείται από ένα σύνολο τεχνητών νευρώνων οι οποίοι επικοινωνούν μεταξύ τους και κάνουν σύνθετους παράλληλους υπολογισμούς.

1.7. Back – Propagation

Όταν χρησιμοποιούμε ένα feed forward neural network η διαδικασία που συμβαίνει είναι η εξής, εισάγουμε ένα διάνυσμα από inputs, η πληροφορία μεταδίδεται προς τα μπροστά σε κάθε τεχνητό νευρώνα και σε κάθε layer και στο τέλος παίρνουμε ένα output. Αυτό ονομάζεται εμπρόσθια διάδοση (forward propagation). Κατά την διάρκεια της εκπαίδευσης του δικτύου καθώς δίνουμε

τιμές από τα inputs και παράγονται τα αντίστοιχα outputs, μετά αυτές τις τιμές των outputs τις βάζουμε στην cost function και τις συγκρίνουμε με τις επιθυμητές τιμές που θα θέλαμε να είχαμε προκειμένου να δούμε κατά πόσο καλά δουλεύει ο αλγόριθμος. Ο αλγόριθμος back – propagation, κάνει το ανάποδο. Ξεκινάει από τη συνάρτηση κόστους και ακολουθεί ακριβώς την ανάποδη κατεύθυνση που χρησιμοποιήθηκε για να υπολογιστεί η cost function με σκοπό να υπολογίσουμε το gradient.

Ο back-propagation ένας από τους σπουδαιότερους αλγορίθμους εφόσον τον χρησιμοποιούμε για να υπολογίζουμε το gradient μιας cost function και ύστερα βάσει αυτού μπορούμε να χρησιμοποιήσουμε gradient descend ή stochastic gradient decsend και για να εκπαιδεύσουμε το τεχνητό νευρωνικό δίκτυο μας. Ο back-propagation αλγόριθμος χρησιμοποιεί τον κανόνα αλυσίδας του απειροστικού λογισμού και υπολογίζει με αυτόν τον τρόπο τις μερικές παραγώγους της cost function ως προς τα weightsκαι τα biases.

Ένα απλό παράδειγμα είναι πως έστω πως έχουμε ένα νευρωνικό δίκτυο που αποτελείται από 3 layers και το κάθε layer έχει έναν νευρώνα. Χ για το input, Β για το hidden layer neuron και Υ για το output layer neuron. Έστω ότι αυτοί οι νευρώνες συνδέονται με αντίστοιχα βάρη w_1 , w_2, όπου w_1 συνδέει το input neuron με το hidden layer neuron και w_2 συνδέει το hidden layer neuron με το output layer neuron. Ο output layer neuron θα μας δώσει κάποια τιμή την οποία θα την συγκρίνουμε με την επιθυμητή τιμή που θα θέλαμε να μας υπολογίζει το δίκτυο. Αυτό θα το κάνουμε χρησιμοποιώντας την cost function, έστω C. Αυτό που θέλουμε εμείς να υπολογίζουμε είναι $\frac{dC}{dw_1}$.Τότε αυτό που θα κάνει ο αλγόριθμος είναι να υπολογίσει την εξής έκφραση:

$$\frac{dC}{dw_1} = \frac{dC}{dY}\frac{dY}{dB}\frac{dB}{dw_1}$$

1.8. Recurrent neural networks

Τα recurrent neural networks είναι μια ιδιαίτερη κατηγορία νευρωνικών δικτύων που συνδέουν και την έννοια του χρόνου στους υπολογισμούς τους. Χρησιμοποιούνται κυρίως σε δεδομένα τα οποία είναι ακολουθίες τιμών, δηλαδή σε δεδομένα όπου η σειρά με την οποία εμφανίζονται οι τιμές έχει σημασία. Έχοντας καταλάβει την έννοια των feed forward networks είναι πολύ εύκολο να καταλάβουμε και τα recurrent networks. Ένα recurrent neural network μπορούμε να το φανταστούμε σαν μία ακολουθία πολλών feedforward neural networks. Έστω ότι έχουμε t χρονικές στιγμές από 1 έως Τ. Τότε μπορούμε να φανταστούμε έναν αλγόριθμο feedforward network (τη χρονική στιγμή 1) να τρέχει βάσει ενός input που του δώσαμε, η πληροφορία να

περνάει μέσα από τα hidden layers και εν τέλει να παράγει ένα output. Αμέσως μετά από αυτόν τον υπολογισμό θέλουμε τώρα να κάνουμε έναν καινούργιο (την χρονική στιγμή t = 2) με ένα καινούργιο inputχρησιμοποιώντας ακριβώς τον ίδιο αλγόριθμο με τις ίδιες παραμέτρους. Θέλουμε όμως επίσης να συμπεριλάβουμε στον υπολογισμό την πληροφορία από την προηγούμενη φορά που τρέξαμε τον αλγόριθμο μιας και στα δεδομένα η σειρά εμφάνισης των τιμών έχει σημασία. Έτσι στον υπολογισμό αυτού του θα λάβουμε υπόψιν μας κάποιο από τα προηγούμενα layersστον υπολογισμό των τωρινών. Για παράδειγμα θα μπορούσε στο παράδειγμα μας το hidden layer την χρονική στιγμή 2 να χρησιμοποιούσε εκτός από το inputπου δώθηκε και τις τιμές που είχε πάρει πριν το hidden layer την χρονική στιγμή 1. Έτσι θα παίρναγε και την προηγούμενη πληροφορία στον τωρινό υπολογισμό το οποίο χαρακτηρίζεται ως ιδιότητα μνήμης. Φυσικά για να συμβεί αυτό θα χρειαζόμασταν βάρη που να συνδέουν τα προηγούμενα hidden layers με τα επόμενα. Αυτά τα βάρη θα ήταν ακριβώς τα ίδια μέσα σε κάθε χρονική στιγμή του υπολογισμού. Αυτό που εξηγήθηκε παραπάνω είναι μια απλή μορφή ενός recurrent neural network (RNN).

Για παράδειγμα μπορούμε να έχουμε ένα κλασσικό δυναμικό σύστημα (deep learning) :

$$S^t = f(s^{(t-1)};\ \theta\ )$$

Όπου S^t καλείται η κατάσταση του συστήματος. Εδώ βλέπουμε καθαρά ότι η κατάσταση την χρονική στιγμή t εξαρτάται από την κατάσταση του συστήματος την χρονική στιγμή t-1 και από παραμέτρους θ. Αυτή η ιδιότητα του συστήματος να εξαρτάται η τωρινή του κατάσταση από την προηγούμενη το κάνει να είναι recurrent.

Με αυτήν την λογική μπορούμε να φτιάξουμε πολλά διαφορετικά recurrent neural networks. Μία από τις πιο συνηθισμένες δομές είναι να έχουμε input, hidden layer, output και κάθε hidden layer να χρησιμοποιεί το input και το hidden layer από το προηγούμενο time step. Έτσι η εξίσωση για το hidden layer την χρονική στιγμή t θα ήταν της μορφής:

$$h^t = f(\ h^{t-1}, x^t; \theta)$$

όπου h^t είναι το hidden layer την χρονική στιγμή t, h^{t-1}είναι το hidden layer την χρονική στιγμή t-1, , x^t είναι το input την χρονική στιγμή tκαι θ είναι το διάνυσμα των παραμέτρων.

Πιο αναλυτικά τρέχοντας ένα τέτοιον αλγόριθμο σε χρόνο t = 1, … , T θα απαιτούσε τον ορισμό του hidden layer στον χρόνο t = 0 και ύστερα την εφαρμογή των παρακάτω εξισώσεων.

$$a^{(t)} = b + w\ h^{(t-1)} + ux^{(t)}$$

$$h^{(t)} = f(a^{(t)}\ )$$

$$o^{(t)} = c + Vh^{(t)}$$

$$y^{(t)} = g(o^{(t)})$$

$$L^{(t)}$$

όπου w είναι ο πίνακας των βαρών από το hidden layer την χρονική στιγμή t-1 στη χρονική στιγμή t, b είναι το bias του hidden layer, u είναι ο πίνακας των βαρών από το input στο hidden layer, f είναι η activation function για του hidden layer, c είναι το bias του output, V είναι ο πίνακας των

βαρών που συνδέει hidden layer με output, καιg είναι η activation function του output.Τέλος, L^t είναι ο παράγοντας της loss function που υπολογίζεται από το συγκεκριμένο χρονικό βήμα. Εν τέλει η loss function θα είναι :

$$L = \sum_{t} L^{(t)}$$

Ο αλγόριθμος που χρησιμοποιείται εδώ για τον υπολογισμό του gradient ονομάζεται back-propagation through time. Δεν έχει κάποια διαφορά από τον γενικό αλγόριθμο back-propagation απλώς διατρέχουμε χρησιμοποιώντας το κανόνα αλυσίδας τον αλγόριθμο προς τα πίσω σε κάθε υπολογιστικό του κόμβο μέσα στον χρόνο. Έστω οτι θέλαμε να υπολογίσουμε $\frac{dL}{dw}$ τότε θα επρέπε να κάνουμε τους υπολογισμούς:

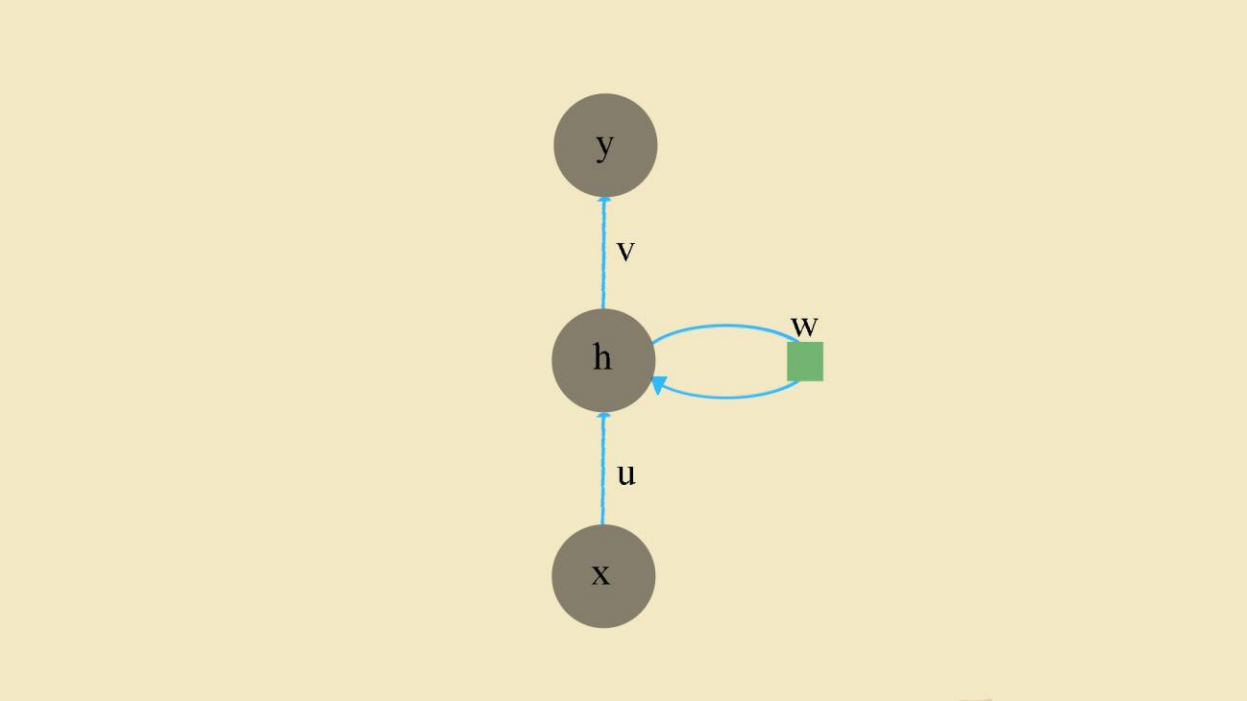
$$\frac{dL}{dw} = \sum_{t=1}^{T} \frac{dL^{(t)}}{dw}$$

όπου,

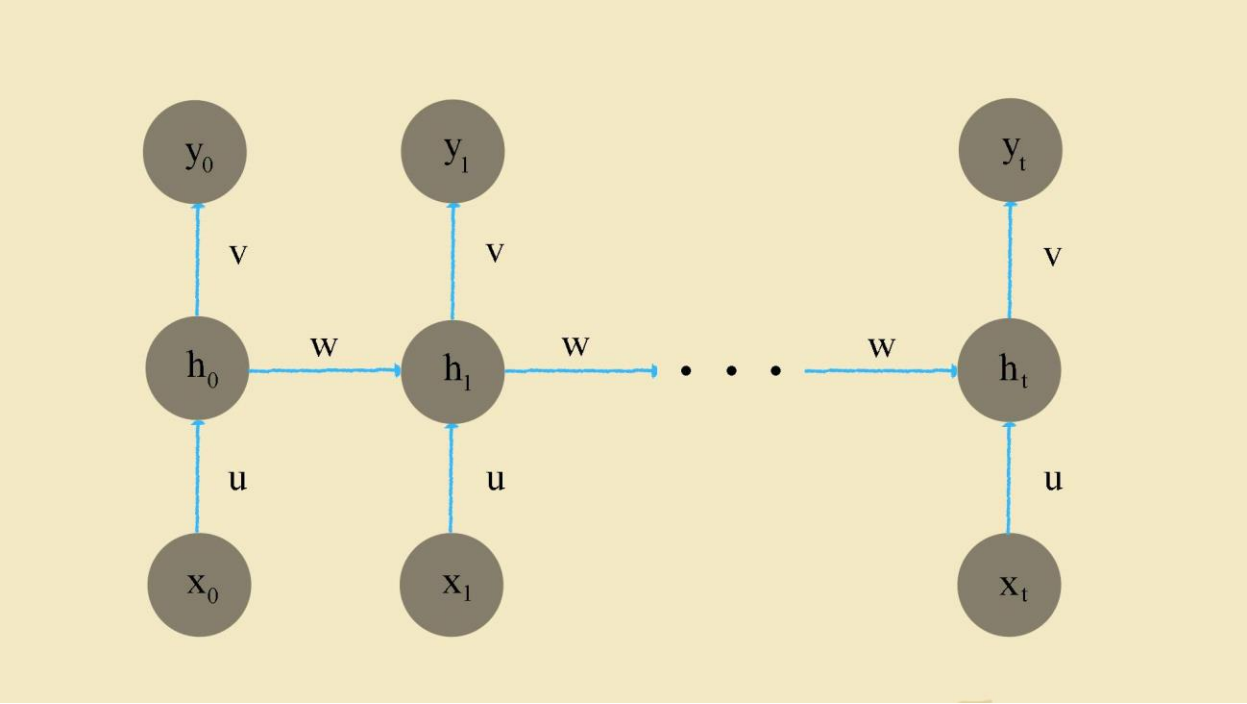
$$\frac{dL^{(t)}}{dw} = \sum_{k=1}^{t} \frac{dL^{(t)}}{dy^{(t)}} \frac{dy^{(t)}}{dh^{(t)}} \frac{dh^{(t)}}{dh^{(k)}} \frac{dh^{(k)}}{dw}$$

Το πρόβλημα τώρα που προκύπτει στην πράξη εφαρμόζοντας αυτόν τον αλγόριθμο είναι ότι κάνοντας πολλούς πολλαπλασιασμούς λόγων των πολλών time steps οι παράγωγοι τείνουν είτε να μηδενίζονται είτε να αυξάνονται απεριόριστα πράγμα που κάνει την διαδικασία της εκπαίδευσης πολύ δύσκολη. Ειδικότερα, όταν ξεκινάμε τον αλγόριθμο και κατευθυνόμαστε σε πολύ παλιές χρονικές στιγμές. Αυτό κάνει την εκπαίδευση μακροχρόνιων εξαρτήσεων με παλαιότερες πληροφορίες που έχει επεξεργαστεί το recurrent network δύσκολη.

Παρακάτω παρατίθενται δύο απεικονίσεις (εικόνα 1.3, εικόνα 1.4) ενός recurrent network,



Εικόνα 1.3. Recurrent neural network



Εικόνα 1.4. Recurrent neural network (διάγραμμα ανοιγμένο στον χρόνο)

1.9. Long-Short Term Memory

Ένα από τα πιο πετυχημένα μοντέλα για την για δεδομένα ακολουθιών είναι το Long Short-Term Memory (LSTM) το οποίο είναι ένα ιδιαίτερο είδος recurrent neural network με μια ιδιαίτερη δομή που το καθιστά ικανό να μπορεί να μαθαίνει μακροχρόνιες σχέσεις μέσα στην δομή των δεδομένων. Το LSTM έχει χρησιμοποιηθεί με μεγάλη επιτυχία σε πολλές εφαρμογές όπως unconstrained handwriting recognition, speech recognition, handwriting generation, machine translation, image captioning, and parsing. Ένα LSTM αποτελείται από 4 μέρη το state, το forget gate, το input gate και το output gate.

Το state είναι το κομμάτι όπου αποθηκεύτε η πληροφορία. Τα υπόλοιπα μέρη λειτουργούν ως σε συνάρτηση με το state και υποδεικνύουν ενεργείς που θα πρέπει να συμβούν. Ένα LSTM μπορεί να ξεχάσει και να αποθηκεύσει καινούργια πληροφορία. Τα gates είναι ένας τρόπος να επιλέξουμε τι πληροφορία θα ξεχαστεί, τι πληροφορία θα εισέλθει και τι πληροφορία θα εξέλθει από το LSTM. Τα gates είναι neural networks με sigmoid activation function.

Η απόφαση το τι θα ξεχαστεί γίνεται από το forget gate. Το forget gate όπως είπαμε και παραπάνω αποτελείται από sigmoid νευρώνες οι οποίοι δέχονται σαν input το τωρινό input και το προηγούμενο output και δίνουν σαν output τιμές από το 0 μέχρι το 1 αναλόγως το αν το νευρωνικό δίκτυο αποφασίζει να κρατήσει ή να ξεχάσει την εκάστοτε τιμή. Δηλαδή μας επιστρέφει ένα διάνυσμα με τιμές από το 0 μέχρι 1. Συνεπώς το output του θα είναι το εξής:

$$f^{(t)} = \sigma(b^f + U^f x^{(t)} + W^f h^{(t-1)})$$

Όπου, $f^{(t)}$ είναι το outputtου forget gate, σ( ) είναι η sigmoid function, b^f το bias διάνυσμα U^f τα βάροι των input, $x^{(t)}$ τα inputτην χρονική στιγμήt, W^f τα βάροι των προηγούμενων outputκαι $h^{(t-1)}$το προηγούμενο output ή το output την χρονική στιγμή t-1(οι εκθέτες (t) συμβολίζουν την χρονική στιγμή, ενώ οι εκθέτες f συμβολίζουν σε ποιο gate είμαστε, f για το forget gate).

Μετά έχουμε το input gate το οποίο ο οποίο αποφασίζει ποιες τιμές θα μεταβάλουμε. Είναι και αυτό ένας νευρωνικό δίκτυο με sigmoid activation function και δέχεται σαν input το $x^{(t)}$ και το $h^{(t-1)}$ και μας δίνει πίσω ένα διάνυσμα με τιμές από 0 μέχρι 1, και μας λέει ποιες τιμές θα προσθέσουμε στο state. Η εξίσωση που μας δίνει το output του είναι :

$$i^{(t)} = \sigma(\, b^i + U^i x^{(t)} + W^i h^{(t-1)})$$

Ύστερα έχουμε τις πιθανές τιμές που μπορεί να αναβαθμίσουμε στο state,

$$C^{(t)} = \sigma(b^c + U^c x^{(t)} + W^c h^{(t-1)})$$

ή

$$C^{(t)} = tanh(b^c + U^c x^{(t)} + W^c h^{(t-1)})$$

Τώρα, έχουμε την εξίσωση που αναβαθμίζει το state :

$$S^{(t)} = f^{(t)} \circ S^{(t-1)} + i^{(t)} \circ C^{(t)}$$

Όπου,$S^{(t)}$το statεστο βήμα τκαι $S^{(t-1)}$ το state στο βήμα t-1. Με το σύμβολο ° συμβολίζεται το Hadamard product το οποίο κάνει την πράξη του πολλαπλασιασμού των αντίστοιχων στοιχείων 2 πινάκων Α και Β ίδιων διαστάσεων και μας επιστρέφει έναν πίνακα της ίδιας διάστασης όπου κάθε στοιχείο του i,j είναι το $a_{i,j}b_{i,j}$ όπου $a_{i,j}$ είναι το i,j στοιχείο του πίνακα Α και $b_{i,j}$ είναι το i,j στοιχείο του Β πίνακα.

Μετά έχουμε το output gate το οποίο διαλέγει ποια στοιχεία από το state θα γίνουν πράγματι output :

$$O^{(t)} = \sigma\left(b^o + U^o x^{(t)} + W^o h^{(t-1)}\right)$$

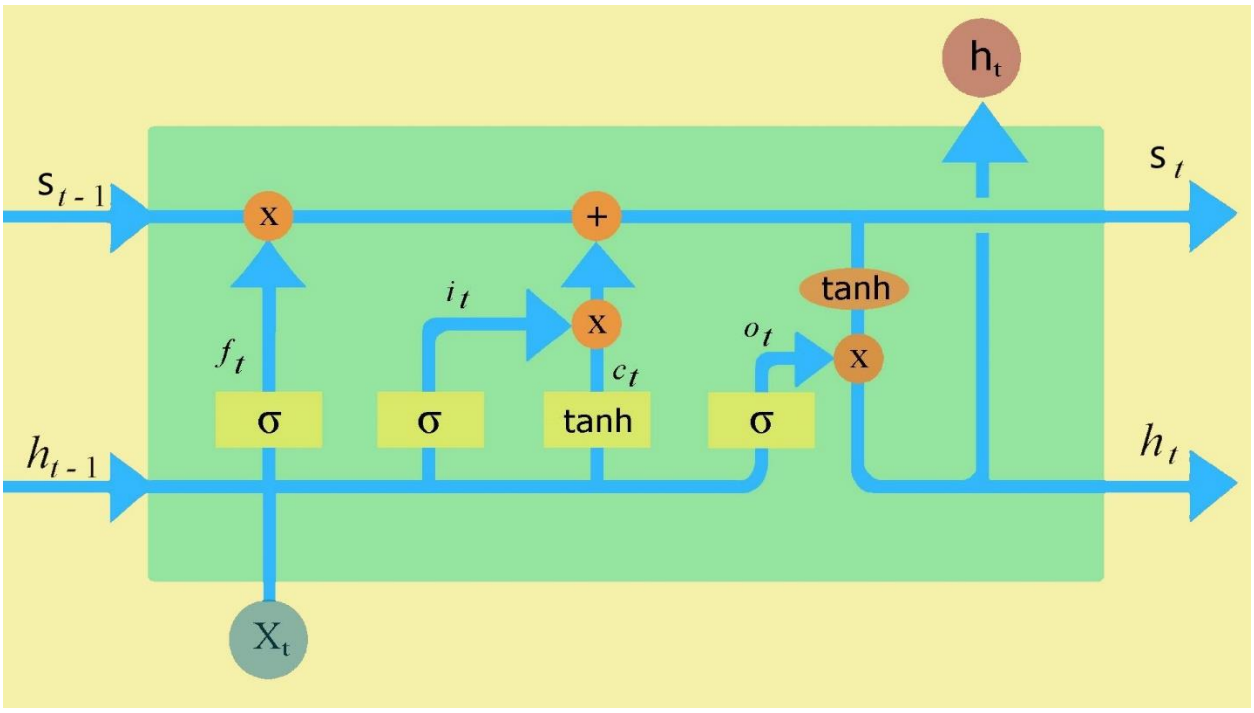
Και τέλος το output,

$$h^{(t)} = O^{(t)} \circ tanh(S^{(t)})$$

Όπου tanh( ) είναι η hyperbolic tangent και δίνεται από τον τύπο :

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

Παρακάτω παρατίθεται μια απεικόνιση ενός LSTM μονάδας(εικόνα 1.5),



Εικόνα 1.5: LSTM unit.

ΚΕΦΑΛΑΙΟ 2

ΜΟΝΤΕΛΑ ΘΝΗΣΙΜΟΤΗΤΑΣ

2.1. Lee-Carter προσέγγιση

Μια από τις πιο γνωστές και ευρέως διαδεδομένη προσέγγιση και μέθοδος για την πρόβλεψη θνησιμότητας είναι η Lee-Carter. Η προσέγγιση αυτή παρουσιάστηκε από τους Ronald D. Lee και Lawrence Carter (1992) σε σχετικό άρθρο με σκοπό να μοντελοποιήσουν και να προβλέψουν την θνησιμότητα στις Ηνωμένες Πολιτείες . Θεωρείται μία από τις σημαντικότερες προσεγγίσεις των τελευταίων δεκαετιών και είναι η κυρίαρχη μέθοδος πρόβλεψης θνησιμότητας. Η μέθοδος αυτή και οι προεκτάσεις της έχουν εφαρμοστεί για την πρόβλεψη θνησιμότητας των G7 χωρών[1] και η υπηρεσία απογραφής των Η.Π.Α χρησιμοποιεί το μοντέλο ως σημείο αναφοράς για τις μακροπρόθεσμες προβλέψεις του προσδόκιμου ζωής.

2.2. Το Μοντέλο 2-παραγόντων (ηλικίας-περιόδου) Lee-Carter

Η ανάλυση τους επικεντρώθηκε στις στοχαστικές μεταβολές μέσα σε ένα μοντέλο μεταβολής της θνησιμότητας . Η διαδικασία τους συνδυάζει ένα λιτό δημογραφικό μοντέλο με την στατιστικής ανάλυσης χρονοσειρών και βασίζεται σε ένα απλό μοντέλο παρελθουσών τάσεων θνησιμότητας ,είναι ορισμένο ως ένας λογαριθμικός μετασχηματισμός του ρυθμού θνησιμότητας ηλικίας (x) , περιόδου (t) εξαρτώμενες από τρείς παραμέτρους

$$\ln(m_{x,t}) = a_x + b_x k_t + \varepsilon_{x,t} \quad ()$$

Με $m_{x,t}$ ο κεντρικός ρυθμός θνησιμότητας στην ηλικία x και χρόνο t

Οι παράμετροι : a_x : ο μέσος όρος (στο χρόνο) της λογαριθμικής θνησιμότητας στην ηλικία x , b_x μετρά τις αντιδράσεις στην x ηλικία για την μεταβολή του συνολικού επιπέδου της θνησιμότητας στο χρόνο , k_t το συνολικό επίπεδο θνησιμότητας σε χρόνο t , και $\varepsilon_{x,t}$ το τυχαίο σφάλμα (κατάλοιπά/σύνολο θορύβων) για τα οποία ισχύει : $\varepsilon_{\chi,t} \overset{iid}{\sim} N(0, V(\varepsilon_{\chi,t}))$

Πρόβλεψη του μοντέλου

Για την πρόβλεψη του μοντέλου υπέθεσαν ότι b_x παραμένει σταθερή στο χρόνο και προβλέπουν τις μελλοντικές τιμές του k_t έπειτα από δοκιμές αρκετών μοντέλων ARIMA (Autoregressive integrated moving average) κατέληξαν στον συμπέρασμα ότι ότι το καταλληλότερο μοντέλο για τα δεδομένα τους ήταν ένας απλός τυχαίος περίπατος με μετατόπιση (Random Walk with Drift) που ορίζεται ως : $k_t = c + k_{t-1} + \varepsilon_t$ όπου c η παράμετρος μετατόπισης και ε_t ο όρος του σφάλματος.

Ο εκτιμητές μεγίστης πιθανοφάνειας του c είναι ο $\hat{c} = \frac{\hat{k}_T - \hat{k}_1}{T-1}$ που εξαρτάται από την πρώτη και την τελευταία παρατήρηση μόνο με διακύμανση $Var(\hat{c}) = \frac{\sigma^2_{rw}}{T-1}$ $\sigma^2_{rw} = \frac{1}{T-1}\sum_{t=1}^{T-1}(\hat{k}_{t+1} - \hat{k}_{t-c})^2$ και $\varepsilon_t \sim N(0, \sigma^2_{rw})$. Για να γίνει πρόβλεψη δυο περιόδους μπροστά $\hat{k}_t = \hat{k}_{t-1} + c + \varepsilon_t = (\hat{k}_{t-2} + c + \varepsilon_{t-1}) + c + \varepsilon_t = \hat{k}_{t-2} + 2c + (\varepsilon_{t-1} + \varepsilon_t)$ αντικαθίσταται η c με την $\hat{c}$ και λαμβάνεται $\hat{k}_t = \hat{k}_{t-2} + 2\hat{c} + (\varepsilon_{t-1} + \varepsilon_t)$

Επομένως για την πρόβλεψη του k_t σε χρόνο $\mathrm{T} + (\Delta t)$ με δεδομένα διαθέσιμα μέχρι την χρονική περίοδο T επαναλαμβάνεται η εξίσωση $\hat{k}_t = \hat{k}_{t-1} + c + \varepsilon_t$ για (Δt) χρονικές στιγμές μπροστά και συνδέεται με εκτίμηση της μετατόπισης: $\hat{k}_{\mathrm{T}+(\Delta t)} = \hat{k}_{\mathrm{T}} + (\Delta t)\hat{c} + \sum_{n=1}^{\Delta t} \varepsilon_{\mathrm{T}+n-1}$ Επειδή οι τυχαίες μεταβλητές ε_t θεωρούνται ανεξάρτητες ο τελευταίος όρος της εξίσωσης κατανέμεται κανονικά με διακύμανση $(\Delta t)\,\sigma^2_{rw}$ και έχει την ίδια κατανομή με την $\sqrt{(\Delta t)}\varepsilon_t$ άρα η εξίσωση μπορεί να γραφεί ως $\hat{k}_{\mathrm{T}+(\Delta t)} = \hat{k}_{\mathrm{T}} + (\Delta t)\hat{c} + \sqrt{(\Delta t)}\varepsilon_t$ η οποία μπορεί να χρησιμοποιηθεί για την άντληση δειγμάτων για την πρόβλεψη σε $\mathrm{T} + (\Delta t)$ χρόνο υπό την πραγματοποίηση του $\hat{k}_1, \ldots, \hat{k}_{\mathrm{T}}$ Εφαρμόζοντας το ίδιο για τις αύξουσες τιμές του (Δt) αποδίδει εξαρτώμενη υπό όρους στοχαστική πρόβλεψη για την $\hat{k}_t$ χρονοσειρά.

Άρα η διακύμανση (υπό όρους) της πρόβλεψης είναι $Var(\hat{k}_{\mathrm{T}+(\Delta t)} \,|\, \hat{k}_1, \ldots, \hat{k}_{\mathrm{T}}) = (\Delta t)\sigma^2_{rw}$

Τα εξαρτώμενα τυπικά σφάλματα της πρόβλεψης αυξάνονται με την τετραγωνική ρίζα της απόστασης του χρονικού ορίζοντα πρόβλεψης.

Αγνοώντας τον όρο του σφάλματος χρησιμοποιούνται σημειακές εκτιμήσεις της στοχαστικής πρόβλεψης οι οποίες ακολουθούν μια ευθεία γραμμή ως συνάρτηση του (Δt) με κλίση c:

$$Var(\hat{k}_{T+(\Delta t)} \left| \hat{k}_1, ..., \hat{k}_T \right.) = \hat{k}_T + (\Delta t)\hat{c}\,.$$

Ακόμη συνδέοντας αυτές τις εκφράσεις στην εμπειρική και διανυσματική εξίσωση είναι δυνατό να γίνει σημειακή πρόβλεψη της λογαριθμικής θνησιμότητας

$$\mu_{T+(\Delta t)} = \bar{m} + \hat{b}\hat{k}_{T+(\Delta t)} = \bar{m} + \hat{b}(\hat{k}_T + (\Delta t)\hat{c})\,.$$

2.3. Το μοντέλο Hatzopoulos-Haberman

Το μοντέλο των Hatzopoulos & Haberman (2009) δίνεται συνοπτικά από τους πιο κάτω τύπους:

$$\log(m_{t,x}) = \mu + \alpha_x + b_t + \sum_{i=1}^{p} d_{i,x} \cdot F_{i,t} + u_{t,x}$$

όπου,

$$\mu = \beta_0 \cdot L_{x,0}$$

$$a_x = \sum_{j=2}^{k} \beta_{j-1} \cdot L_{x,j-1}\ ,$$

$$d_{i,x} = k \sum_{j=2} e_{j,i} \cdot L_{x,j-1}\ ,$$

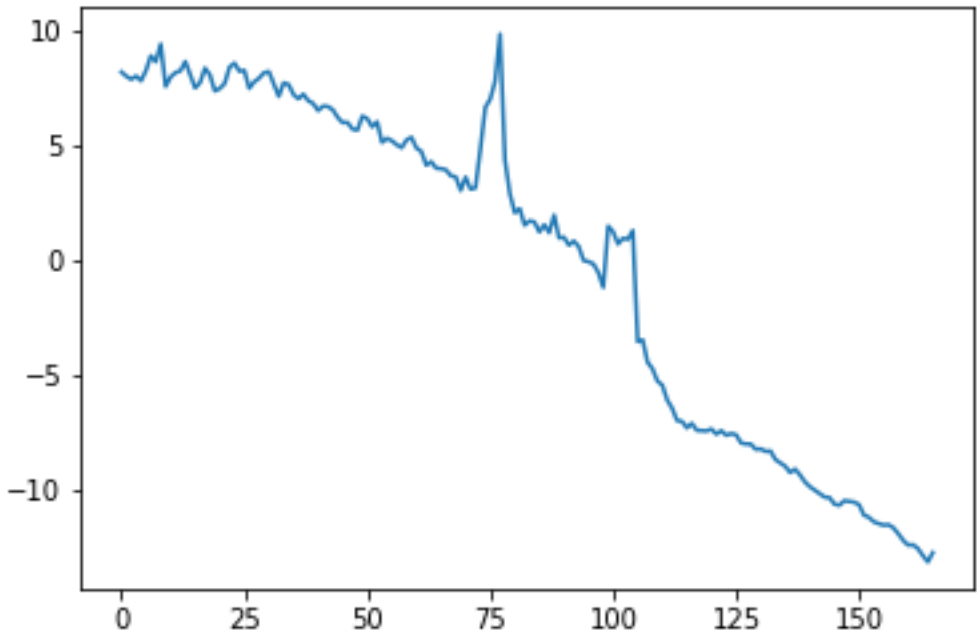
και $\quad b_t = \sum_{i=1}^{p} m_i \cdot F_{i,t}$ για $m_i = L_{x,0} \cdot e_{1,i}$.

Το μοντέλο αυτό μπορεί να θεωρηθεί τώρα ως μέλος της κλάσης των log-linear models, με την ηλικία και τον χρόνο να είναι οι δύο βασικές τυχαίες επιδράσεις, παρόμοιο με το association model του Goodman (1991). Το μ είναι ο ολικός μέσος, το α_x και το b_t είναι η κύρια επίδραση της ηλικίας και η κύρια επίδραση του χρόνου αντίστοιχα. Το $d_{i,x}$ μας δείχνει ποιες είναι οι στατιστικά σημαντικές ηλικίες που αντιστοιχούν στα $F_{i,t}$, όπου $F_{i,t}$ τα principal components scores και $u_{t,x}$ ο διαταρακτικός όρος. Το b_t είναι ένας δείκτης της θνησιμότητας σε συνάρτηση με τον χρόνο που περιγράφει την γενική τάση ως προς τον χρόνο της log θνησιμότητας σε όλες τις ηλικίες (Hatzopoulos and Haberman (2009)) .

ΚΕΦΑΛΑΙΟ 3

ΜΙΑ ΕΦΑΡΜΟΓΗ ΣΤΟΝ ΑΝΑΛΟΓΙΣΜΟ

Θέλουμε να χρησιμοποιήσουμε το παραπάνω μοντέλο για να εκτιμήσουμε τη θνησιμότητα στην Αγγλία και στην Ουαλία. Οι παρατηρήσεις που έχουμε είναι από το 1841 έως το 2006 για ηλικίες από 0 έως 89. Εμείς αυτό που θέλουμε να κάνουμε είναι να εκτιμήσουμε μελλοντικές τιμές του δείκτη $F_{1,t}$ ώστε να μπορέσουμε να κάνουμε προβλέψεις σύμφωνα με το παραπάνω μοντέλο. Η χρονοσειρά είναι η εξής:

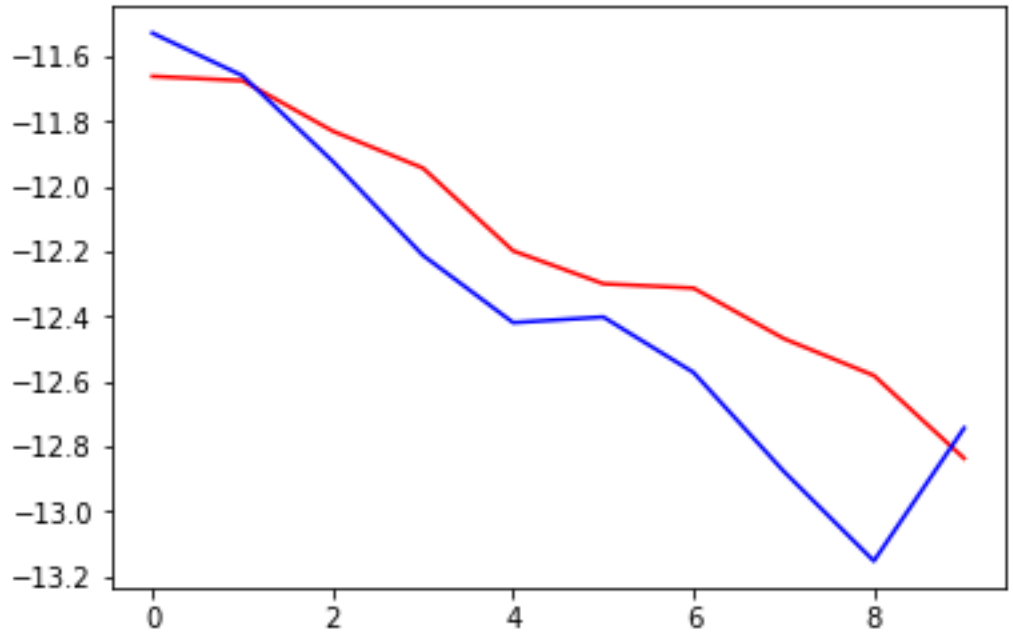


Εικόνα 3.1. Ο δείκτης $F_{1,t}$

Αυτό που θέλουμε να κάνουμε είναι να συγκρίνουμε τον ποιο συνηθισμένο τρόπο που χρησιμοποιείται στα αναλογιστικά μαθηματικά, τα μοντέλα Arima με ένα νευρωνικό δίκτυο που αποτελείται από LSTM μονάδες. Για αυτόν τον λόγο αφαιρούμε τις 10 τελευταίες παρατηρήσεις από τις συνολικά 166 παρατηρήσεις και εκτιμούμε τις τελευταίες 10 με βάση την πληροφόρησή μας από τις προηγούμενες 156.

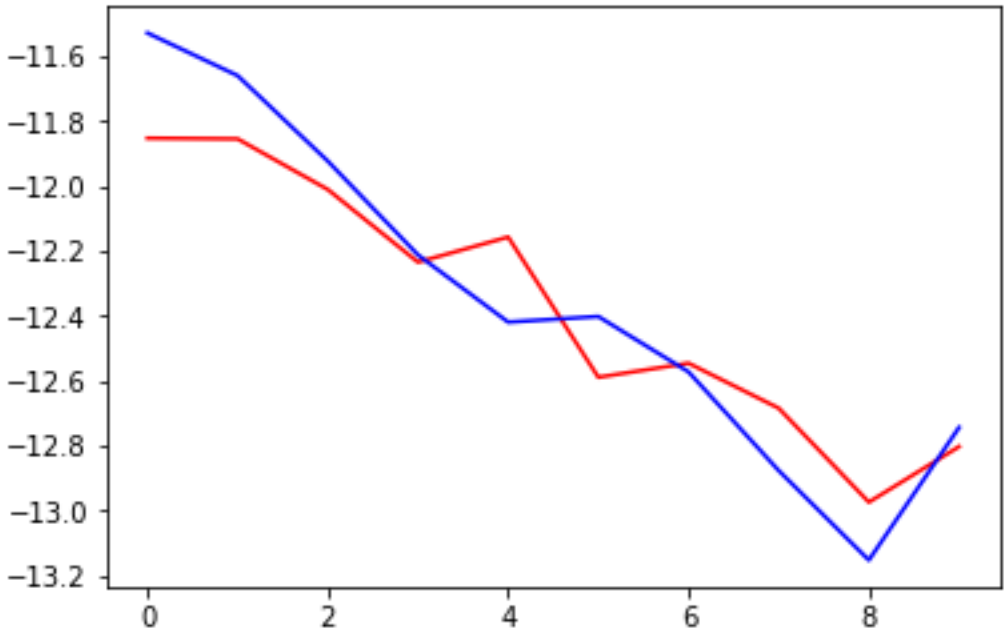
Με βάση τα μοντέλα Arima το καταλληλότερο μοντέλο που επιτυγχάνει να περιγράψει ικανοποιητικά τη διαδικασία είναι ένα SARIMA $(0, 1, 1)(0, 1, 1)_5$ δηλαδή ένα μοντέλο ARIMA (0,1,1) με εποχικότητα της μορφής (0,1,1) τάξης 5.

Η πρόβλεψη που πετυχαίνει το μοντέλο SARIMA $(0, 1, 1)(0, 1, 1)_5$ έχει mean squared error (MSE) 0.07241695340219977 και δίνεται διαγραμματικά ως ακολούθως:



Εικόνα 3.2. Πρόβλεψη δείκτη με το μοντέλο SARIMA$(0, 1, 1)(0, 1, 1)_5$ (με μπλε τα πραγματικά δεδομένα).

Το νευρωνικό δίκτυο αποτελείται από το διάνυσμα των 5 inputs, ένα layer με 20 lstm, ένα layer των 50 lstm, ένα layer των 100 lstm και ένα layer των 5 νευρώνων με activation function την linear. Ο optimizer που χρησιμοποιήθηκε είναι rmsprop. Το batch size ήταν 5. Η loss function που χρησιμοποιήθηκε είναι το MSE. Μετά από 350 epochs επιτυγχάνει διαγραμματικά την πιο κάτω πρόβλεψη με MSE=0.03271191843465604.



Εικόνα 3.3. Πρόβλεψη δείκτη με νευρωνικά δίκτυα (με μπλε τα πραγματικά δεδομένα)

Σχόλια:

Όπως φαίνεται από τα 2 παραπάνω γραφήματα το νευρωνικό δίκτυο σύμφωνα με το MSE είναι πολύ καλύτερο στην πρόβλεψη από το βέλτιστο ARIMA model.

Βιβλιογραφία

A. Graves, 2012, Supervised Sequence Labelling with Recurrent Neural Networks, Technische Universitat Munchen.

I. Goodfellow, Y. Benjio, A. Courville, 2016. Deep Learning. The MIT Press. ISBN**:** 9780262035613.

P. Hatzopoulos, S. Haberman, 2009. A parameterized approach to modeling and forecasting mortality. Insurance: Mathematics and Economics 44, 103 – 123.

P. Hatzopoulos, S. Haberman, 2011. A dynamic parameterization modeling for the age-period-cohort mortality. Insurance: Mathematics and Economics 49, 155 – 174.

R. Lee, L. Carter, 1992. Modeling and forecasting U.S. mortality. Journal of the American Statistical Association 87, 659–671.

T. Mitchell, (1997). *Machine Learning*. McGraw Hill. p. 2. ISBN 0-07-042807-7.

M. Mohri, A. Rostamizadeh, A. Talwalkar, (2012). *Foundations of Machine Learning*. USA, Massachusetts: MIT Press. ISBN 9780262018258.

M. A. Nielsen, 2015. Neural Networks and Deep Learning", Determination Press.

A. Samuel (1959). "Some Studies in Machine Learning Using the Game of Checkers". *IBM Journal of Research and Development*. **3** (3): 210–229. doi:10.1147/rd.33.0210.

A. L. Samuel (1988). "Some Studies in Machine Learning Using the Game of Checkers. I". *Computer Games I*. Springer, New York, NY. pp. 335–365. doi:10.1007/978-1-4613-8716-9_14. ISBN 9781461387183.

I. Sutskever, 2013, Training recurrent neural networks, University of Toronto.

P.-N. Tan, M. Steinbach, V. Kumar, 2005. Introduction to Data Mining, Pearson.