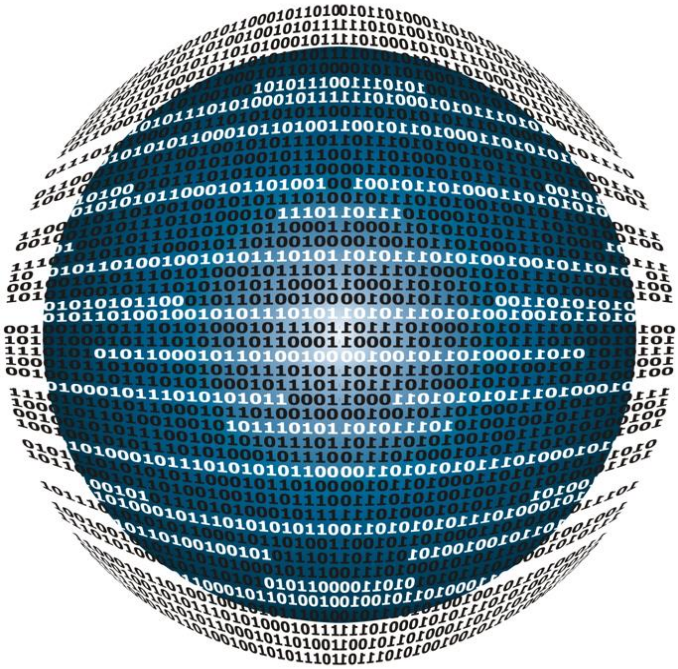
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΑΘΗΜΑ : ΥΠΟΛΟΓΙΣΤΙΚΗ ΝΟΗΜΟΣΥΝΗ

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 2019-2020
ΜΕΡΟΣ Α΄



Εργασία του Φοιτητή

Θωμά Χατζόπουλου
ΑΜ: 1054288

chatzothomas@gmail.com
up1054288@upnet.gr

Διδάσκοντες: Σ. Λυκοθανάσης, Δ. Κουτσομητρόπουλος

ΠΑΤΡΑ | Μάρτιος - Απρίλιος 2020

Περιεχόμενα

Α. Συνεργατικό Φιλτράρισμα με Χρήση Νευρωνικών Δικτύων για Συστάσεις Ταινιών

Εισαγωγή

α. Σκοπός άσκησης

Στην παρούσα εργασία εξετάζω τη χρήση ενός πολυεπίπεδου ΤΝΔ για την πρόβλεψη αξιολογήσεων χρηστών σε ταινίες με εφαρμογή συνεργατικού φιλτραρίσματος. Πιο συγκεκριμένα, το ζητούμενο της εργασίας είναι να κατασκευαστεί και να εκπαιδευτεί ένα ΤΝΔ που θα δέχεται στην είσοδό του ένα χρήστη και θα παράγει στην έξοδο τις αξιολογήσεις για όλες τις ταινίες ενός συγκεκριμένου συνόλου δεδομένων. Κατά τη διάρκεια των πειραμάτων χρησιμοποιούνται μετρικές για την αξιολόγηση των μοντέλων, όπως η Ρίζα του Μέσου Τετραγωνικού Σφάλματος (RMSE) και το Μέσο Απόλυτο Σφάλμα (MAE).

β. Σύνολο δεδομένων

Για το σκοπό αυτό αξιοποίησα, όπως ζητήθηκε, το σύνολο δεδομένων "**MovieLens 100K**[1]**"** που περιλαμβάνει περίπου 100k αξιολογήσεις ταινιών από N = 943 χρήστες για M = 1682 ταινίες. Τα δεδομένα των αξιολογήσεων που χρησιμοποίησα περιέχονται στο αρχείο u.data του παραπάνω συνόλου δεδομένων. Τα δεδομένα στο αρχείο είναι χωρισμένα κατά στήλες με tab, με τα ονόματα των στηλών να είναι:

user id | item id | rating | timestamp

Τα δεδομένα που μας είναι απαραίτητα στην παρούσα εργασία είναι αυτά των 'user id', που περιέχει τον κωδικό χρήστη, 'item id', που περιέχει τον κωδικό της ταινίας που βαθμολογεί ο χρήστης και 'rating', που περιέχει την βαθμολογία που βάζει ο χρήστης. Τα δεδομένα της στήλης 'timestamp' στα πλαίσια των πειραμάτων που πραγματοποιούνται δεν είναι απαραίτητα.

γ. Γλώσσα και Περιβάλλον Υλοποίησης, βιβλιοθήκες που χρησιμοποιήθηκαν

Για την υλοποίηση της άσκησης χρησιμοποίησα τη γλώσσα προγραμματισμού Python:

- Πρόγραμμα συγγραφής και εκτέλεσης: **PyCharm 2020.1**[2]
- Python interpreter: **Python 3.8.2**[3]

Βιβλιοθήκες (python) που χρησιμοποιήθηκαν:
- **numpy**
- **pandas**
- από την sklearn.model_selection την **Kfold**
- **keras**
- από την Keras την **backend**
- από την Keras.layers την **Dense**
- από την Keras.callback την **EarlyStopping**
- από την keras.regularizers την **l1**
- **matplatlib.pyplot**

δ. link κώδικα

Τα αρχεία με τους κώδικες που χρησιμοποιήθηκαν βρίσκονται σε φάκελο στο παρακάτω σύνδεσμο:
https://drive.google.com/open?id=18wUQ2TT-pBYUQBO_w7XvCveOnkZGBtdR

[1] https://grouplens.org/datasets/movielens/100k/
[2] https://www.jetbrains.com/pycharm/
[3] https://www.python.org/downloads/release/python-382/

Α1. Προεπεξεργασία και Προετοιμασία δεδομένων

α. Κεντράρισμα (centering)

Οι τιμές των αξιολογήσεων των χρηστών του συνόλου δεδομένων είναι ακέραιες και κινούνται στο διάστημα [1,5]. Ωστόσο, υπάρχουν περιπτώσεις χρηστών των οποίων οι αξιολογήσεις εμφανίζουν κάποιο βαθμό πόλωσης.

Η μέθοδος που χρησιμοποίησα στον αλγόριθμο για να μετριαστεί αυτό το φαινόμενο είναι το κεντράρισμα (centering) των δεδομένων και επιτυγχάνεται αφαιρώντας το μέσο όρο των αξιολογήσεων ενός χρήστη από όλες τις βαθμολογίες που έχει δώσει. Το κεντράρισμα των δεδομένων δεν είναι απαραίτητο για την διεξαγωγή του πειράματος, αλλά όταν πραγματοποιείται αφενός επηρεάζονται αρκετές παράμετροι και λειτουργίες, όπως ο τρόπος συμπλήρωσης των ελλιπών τιμών ή η επιλογή της συνάρτησης ενεργοποίησης, αφετέρου δε, τα αποτελέσματα που λαμβάνουμε είναι πολύ καλύτερα.

Το νέο διάστημα στο οποίο κινούνται οι τιμές είναι το [-3.7241, 3.5080]. Πλέον δεν υπάρχει πόλωση στα δεδομένα, αλλά κάθε διάνυσμα έχει την ίδια αξία.

β. Ελλιπείς τιμές

Ένα από τα συχνά προβλήματα που συναντάται συχνά σε σύνολα δεδομένων είναι η έλλειψη τιμών σε κάποιες εγγραφές-διανύσματα. Πιο συγκεκριμένα για το σύνολο δεδομένων “MovieLens 100K[1]”, δεν υπάρχουν αξιολογήσεις όλων των χρηστών για όλες τις ταινίες, επομένως πολλά διανύσματα αξιολόγησης είναι ελλιπή, με αποτέλεσμα η εκπαίδευση να μην μπορεί να λειτουργήσει, όταν απαντώνται τέτοιες τιμές. Υπάρχουν διάφοροι και άμεσοι τρόποι για να συμπληρώσει κάποιος τις ελλιπείς τιμές των διανυσμάτων, όπως αυτές που προτείνονται:

i. μια τυχαία επιλεγμένη τιμή στο κατάλληλο εύρος
ii. μηδέν
iii. με τον μέσο όρο του διανύσματος αξιολόγησης

Της συμπλήρωσης των ελλιπών τιμών **προηγήθηκε το κεντράρισμα των δεδομένων**.

i. Έχοντας μόνο τα στοιχεία που δίνονται για το σύνολο δεδομένων μπορούμε να υπολογίσουμε ότι κατά μέσο όρο κάθε χρήστης έχει αξιολογήσει $\text{Μ. Ο.}_{\text{ΤαινιώνΧρήστη}} = \frac{αριθμός\_αξιολογήσεων}{N} = \frac{100.000}{943} \approx 106$ ταινίες από τις M = 1682 ταινίες που περιέχει συνολικά το σύνολο δεδομένων, δηλαδή μόλις το $\frac{\text{Μ.Ο.}_{\text{ΤαινιώνΧρήστη}}}{M} = \frac{106,04}{1682} \approx 6,3\%$ των ταινιών (δεν έχει ακόμα εξετάσει το ενδεχόμενο αν υπάρχει κάποια ταινία χωρίς καμία αξιολόγηση). Επιλέγοντας, συνεπώς, μια τυχαία τιμή στο επιτρεπόμενο εύρος, ουσιαστικά συμπληρώνουμε το 93,7% των αξιολογήσεων του κάθε χρήστη με μια τυχαία τιμή, κάτι το οποίο δεν είναι επιθυμητό, καθώς ουσιαστικά ακυρώνει τις υπόλοιπες αξιολογήσεις του χρήστη, αφού το μεγαλύτερο βάρος κατά την εκπαίδευση θα έχουν οι τυχαίες τιμές και όχι πραγματικές αξιολογήσεις που ήδη έχει πραγματοποιήσει ο χρήστης. **Αυτή η επιλογή συμπλήρωσης ελλιπών τιμών απορρίπτεται**.

ii. Αφού ήδη έχει προηγηθεί κεντράρισμα των δεδομένων, η συμπλήρωση των ελλιπών τιμών με μηδέν φαίνεται να είναι μια καλή επιλογή, καθώς τώρα το μηδέν αντιστοιχίζεται στο μέσο όρο των αξιολογήσεων που έχει δώσει ο χρήστης. Συνεπώς, αυτή είναι μια καλή επιλογή, εφόσον έχει προηγηθεί το κεντράρισμα των δεδομένων.

iii. Αφού ήδη έχει προηγηθεί κεντράρισμα των δεδομένων, η συμπλήρωση των ελλιπών τιμών με το μέσο όρο του διανύσματος αξιολόγησης θα προκαλέσει ανομοιομορφία στο σύνολο δεδομένων, καθώς όσες ταινίες έχει ήδη αξιολογήσει ο χρήστης είναι κεντραρισμένες, ενώ οι νέες τιμές που συμπληρώνουμε δεν θα είναι. Συνεπώς, η συμπλήρωση των ελλιπών με το μέσο όρο του διανύσματος αξιολόγησης είναι μια καλή επιλογή αν δεν έχει προηγηθεί κεντράρισμα.

Όπως γίνεται φανερό και από την παραπάνω ανάλυση, για την συμπλήρωση των ελλιπών τιμών **επέλεξα τη δεύτερη μέθοδο (συμπλήρωση με 0)**, που ουσιαστικά **μετά το κεντράρισμα** αντιστοιχεί σε συμπλήρωση με το μέσο όρο του διανύσματος αξιολόγησης.

γ. Κανονικοποίηση (rescaling)

Με βάση την προτεινόμενη αρχιτεκτονική του ΤΝΔ, στην έξοδο αναμένονται τιμές αξιολογήσεων για έναν συγκεκριμένο χρήστη. Είναι προφανές ότι η συνάρτηση ενεργοποίησης στο επίπεδο εξόδου θα πρέπει να είναι σε θέση να παράξει τιμές σε αυτό το διαθέσιμο εύρος. Προκειμένου αυτό να είναι εφικτό χρειάζεται τα δεδομένα να κανονικοποιηθούν σε διάστημα τιμών κατάλληλο για τη συγκεκριμένη συνάρτηση ενεργοποίησης.

Πιο συγκεκριμένα, αφού εφάρμοσα κεντράρισμα στα δεδομένα και συμπλήρωσα τις ελλιπείς τιμές των διανυσμάτων αξιολόγησης ταινιών των χρηστών, **εντόπισα τη μέγιστη κατά απόλυτη τιμή αξιολόγησης για τον κάθε χρήστη ξεχωριστά και στη συνέχεια διαίρεσα με την τιμή αυτή το διάνυσμα αξιολόγησης του εκάστοτε χρήστη,** προκειμένου τα δεδομένα να κανονικοποιηθούν και να ανήκουν στο επιθυμητό εύρος [-1, 1]**.**

δ. Διασταυρούμενη Επικύρωση (cross-validation)

Προκειμένου να εκπαιδεύσω το ΤΝΔ και να επιβεβαιώσω την ορθή λειτουργία του χρειάζεται ο αλγόριθμος να διαχωρίσει τα δεδομένα του συνόλου δεδομένων σε σύνολα εκπαίδευσης και ελέγχου. Για το λόγο αυτό για όλα τα πειράματα χρησιμοποιώ 5-fold Cross-Validation. Με αυτόν τον τρόπο, σε κάθε εκπαίδευση χρησιμοποιώ τα 4/5 του συνόλου δεδομένων για εκπαίδευση και το υπόλοιπο 1/5 για τον έλεγχο. Σε κάθε επόμενη εκπαίδευση τα επιμέρους σύνολα δεδομένων εναλλάσσονται μεταξύ τους, ώστε όλα να χρησιμοποιηθούν και στη φάση της εκπαίδευσης, αλλά και στη φάση του ελέγχου.

Α2. Επιλογή αρχιτεκτονικής

Όσον αφορά την τοπολογία των ΤΝΔ για την εκπαίδευση τους με τον Αλγόριθμο Οπισθοδιάδοσης του Σφάλματος (back-propagation) χρησιμοποίησα ΤΝΔ με ένα κρυφό επίπεδο και πειραματίστηκα με τον αριθμό των κρυφών κόμβων. Για την εκπαίδευση του δικτύου χρησιμοποίησα ως αρχικό ρυθμό μάθησης η = 0.001, ενώ σε επόμενες εκπαιδεύσεις, όπως είναι και ζητούμενο, ο ρυθμός μάθησης αυξήθηκε μέχρι και η=0.1.

α. Μετρικές

Η αξιολόγηση των μοντέλων για τις βαθμολογίες που περιέχονται στα σύνολα ελέγχου μπορεί να πραγματοποιηθεί με τη Ρίζα του Μέσου Τετραγωνικού Σφάλματος (RMSE), καθώς και με το Μέσο Απόλυτο Σφάλμα (MAE).

Ρίζα μέσου τετραγωνικού σφάλματος (RMSE)

Ισχύει[4] ότι:

> *Αν* Y_t *η παρατήρηση τη χρονική στιγμή* t, F_t *η πρόβλεψη του* Y_t *και* e_t *το σφάλμα που προκύπτει από τη σχέση* $e_t = Y_t - F_t$, *τότε*:
>
> - Μέσο τετραγωνικό σφάλμα (MSE): $mean(e_t^2) = mean(Y_t - F_t)^2$
> - Ρίζα μέσου τετραγωνικού σφάλματος (RMSE): $\sqrt{MSE} = \sqrt{mean(Y_t - F_t)^2}$

Η ρίζα του μέσου τετραγώνου σφάλματος αντιπροσωπεύει την τετραγωνική ρίζα της δεύτερης στιγμής δειγματοληψίας των διαφορών μεταξύ των προβλεπόμενων τιμών και των παρατηρούμενων τιμών ή του τετραγωνικού μέσου αυτών των διαφορών. Χρησιμεύει, κυρίως, για τη συγκέντρωση των μεγεθών των σφαλμάτων στις προβλέψεις για διάφορους χρόνους σε ένα μόνο μέτρο της προβλεπτικής ισχύος[4].

Η ρίζα του μέσου τετραγωνικού σφάλματος είναι ένα μέτρο ακρίβειας, για να συγκρίνει τα σφάλματα πρόβλεψης διαφορετικών μοντέλων για ένα δεδομένο σύνολο δεδομένων, όπως εδώ χρησιμοποιούμε το "MovieLens 100K[1]" και όχι για σύνολα δεδομένων, καθώς εξαρτάται από την κλίμακα.

Η RMSE είναι πάντοτε μη αρνητική και μια τιμή 0 (σχεδόν ποτέ δεν επιτυγχάνεται στην πράξη, για αυτό και δεν μπορεί πάντα να χρησιμοποιείται ως κριτήριο τερματισμού) θα έδειχνε την τέλεια εφαρμογή στα δεδομένα. Γενικά, μία χαμηλότερη τιμή της ρίζας του μέσου τετραγωνικού σφάλματος είναι καλύτερη από μία υψηλότερη, καθώς δηλώνει ότι πετυχαίνουμε μικρότερο σφάλμα. Η τιμή του Μέσου Τετραγωνικού Σφάλματος (MSE) μπορεί ακόμα και να ξεπεράσει τη μονάδα, αλλά η Ρίζα του Μέσου Τετραγωνικού Σφάλματος δεν θα ξεπερνάει τη μονάδα (στη γενική περίπτωση). Αυτό μπορεί να συμβεί όταν οι τιμές που εξετάζονται είναι μεγάλης τάξης μεγέθους αυτό και τότε δεν αποτελεί πρόβλημα, ενώ το μοντέλο εξακολουθεί να είναι ακριβές. Όταν, όμως οι τιμές των δεδομένων κυμαίνονται, όπως εδώ σε εύρος [-1,1], τιμή RMSE μεγαλύτερη την μονάδας δεν είναι αποδεκτή για το μοντέλο. Όταν η τιμή του σφάλματος είναι μικρότερη του μηδενός και υπολογίζουμε το Μέσο Τετραγωνικό Σφάλμα (MSE), τότε η Ρίζα του Μέσου Τετραγωνικού Σφάλματος αρχίζει να προσεγγίζει τη μονάδα.

[4] Hyndman, Rob J.; Koehler, Anne B. (2006). "Another look at measures of forecast accuracy". International Journal of Forecasting. 22 (4): 679–688. doi:10.1016/j.ijforecast.2006.03.001 (https://www.sciencedirect.com/science/article/pii/S0169207006000239)

Έτσι και εδώ για τα διάφορα μοντέλα νευρωνικών δικτύων που εξετάζονται, η μετρική RMSE είναι ενδεικτική για το πόσο καλό είναι το συγκεκριμένο μοντέλο που εξετάζεται σε σύγκριση με τα υπόλοιπα. Κάθε μοντέλο διαφέρει από τα υπόλοιπα στο πλήθος των νευρώνων, όταν μεταβάλλεται ο αριθμός των κόμβων του κρυφού επιπέδου, στο ρυθμό μάθησης και τη σταθερά της ορμής, ενώ παραμένει σταθερό το μέγεθος των διανυσμάτων εισόδου και εξόδου, καθώς κάθε φορά εξετάζεται το ίδιο σύνολο δεδομένων. Συνεπώς, **όσο μικρότερη τιμή RMSE έχει ένα μοντέλο, τόσο καλύτερο** είναι και αυτό είναι επιθυμητό και με το συγκεκριμένο σύνολο δεδομένων.

Η επίδραση κάθε σφάλματος στη ρίζα του μέσου τετραγωνικού σφάλματος είναι ανάλογη με το μέγεθος του τετραγώνου σφάλματος. Έτσι τα μεγαλύτερα σφάλματα έχουν δυσανάλογα μεγάλη επίδραση στη ρίζα του μέσου τετραγωνικού σφάλματος. Κατά συνέπεια, η ρίζα του μέσου τετραγωνικού σφάλματος είναι ευαίσθητη στις υπερβάσεις, για αυτό πολλές φορές προτιμάται η μετρική του Μέσου Απολύτου Σφάλματος (MAE), η οποία αναλύεται παρακάτω.

Μέσο Απόλυτο Σφάλμα (MAE)

Ισχύει[4] ότι:

> *Αν* Y_t *η παρατήρηση τη χρονική στιγμή* t, F_t *η πρόβλεψη του* Y_t *και* e_t *το σφάλμα που προκύπτει από τη σχέση* $e_t = Y_t - F_t$, *τότε*:
>
> $$\text{Μέσο Απόλυτο Σφάλμα } (MAE): mean(|e_t|) = mean(|Y_t - F_t|)$$

Ουσιαστικά, στο μέσο απόλυτο σφάλμα παίρνουμε τη μέση τιμή του απολύτου σφάλματος, δηλαδή της διαφοράς της πραγματικής από τη προβλεπόμενη τιμή κατά μήκος των δυο δειγμάτων εκπαίδευσης και πρόβλεψης και παίρνουμε τα αντίστοιχα μέσα απόλυτα σφάλματα εκπαίδευσης και πρόβλεψης· το μέσο απόλυτο σφάλμα είναι ο μέσος όρος των απόλυτων τιμών των σφαλμάτων.

Το μέσο απόλυτο σφάλμα είναι θεμελιωδώς ευκολότερο στην κατανόησή του από την τετραγωνική ρίζα του μέσου τετραγωνικού σφάλματος. Επιπλέον, κάθε σφάλμα επηρεάζει το μέσο απόλυτο σφάλμα σε άμεση αναλογία με την απόλυτη τιμή του σφάλματος, πράγμα που δεν συμβαίνει με τη ρίζα του μέσου τετραγωνικού σφάλματος.

Προφανώς, όπως και με το RMSE, κατά την εκπαίδευση του νευρωνικού με το σύνολο δεδομένων **είναι επιθυνητό το MAE να είναι ελάχιστο, αν όχι και 0**. Παρακάτω παρατίθεται πίνακας σύγκρισης [5]των χαρακτηριστικών των τιμών RMSE και MAE:

Characteristic	RMSE	MAE
1. Ability to budget components of disagreement and agreement	Yes	Yes
2. Commutative property for components of disagreement	Yes	Yes
3. Commutative property for components of agreement	No	No
4. Unique solution for minimum deviation	Yes	No
5. Sensitivity to outliers	More	Less
6. Sensitivity to change of resolution	More	Less
7. Interpretable in terms of moving mass	No	Yes
8. Consistent with categorical case	No	Yes

[5] Pontius, Robert; Thontteh, Olufunmilayo; Chen, Hao (2008). "Components of information for multiple resolution comparison between maps that share a real variable". Environmental Ecological Statistics. 15: 111–142 (https://link.springer.com/content/pdf/10.1007/s10651-007-0043-y.pdf)

β. Μέγεθος και κωδικοποίηση διανύσματος εισόδου

Δεδομένου ότι μια είσοδος στο ΤΝΔ πρέπει να αναπαριστά έναν χρήστη και θεωρώντας ότι όλοι οι χρήστες αρχικά είναι ισοδύναμοι, ένας χρήστης μπορεί να αναπαρασταθεί από ένα δυαδικό διάνυσμα N = 943 θέσεων, όσοι και οι χρήστες του συνόλου δεδομένων, δηλαδή ακολουθώντας την τεχνική one-hot encoding (τιμή 1 στη θέση που αντιστοιχεί στο χρήστη και 0 αλλού[6]).

Όταν χρησιμοποιείται Binary ή κατά Gray κωδικοποίηση, απαιτείται αποκωδικοποιητής για τον προσδιορισμό της κατάστασης, κάτι που δεν χρειάζεται με την one-hot κωδικοποίηση, καθώς κάθε κατάσταση προσδιορίζεται μοναδικά από το διάνυσμα εισόδου (υπάρχει η τιμή 1 μόνο στο χρήστη που εξετάζεται, ενώ όλες οι άλλες θέσεις είναι 0).

Συνεπώς, το διάνυσμα εισόδου είναι **δυαδικό, N = 943 θέσεων (όσοι και οι χρήστες), με κωδικοποίηση one-hot**.

γ. Μέγεθος διανύσματος εξόδου

Δεδομένου ότι η έξοδος πρέπει να αναπαριστά τις αξιολογήσεις του χρήστη για όλες τις ταινίες, το διάνυσμα εξόδου, όπως και οι **νευρώνες τις εξόδου**, με όμοια λογική που κωδικοποιήσαμε το διάνυσμα εισόδου, είναι ένα **διάνυσμα μεγέθους M = 1682 θέσεων, όσο το πλήθος των ταινιών**, κάθε μια από τις θέσεις του οποίου αντιστοιχίζεται μοναδικά σε κάθε ταινία.

δ. Συνάρτηση ενεργοποίησης κρυφών κόμβων

Ένα από τα σημαντικά μέρη του σχεδιασμού ενός τεχνητού νευρωνικού δικτύου είναι η επιλογή κατάλληλης συνάρτηση ενεργοποίησης για τους κρυφούς κόμβους. Στα συγκεκριμένα πειράματα, προς το παρόν, αρκεί ένα κρυφό επίπεδό (αργότερα εξετάζεται δίκτυο με περισσότερα κρυφά επίπεδα), όπως αναλύεται και παρακάτω (βλέπε παράγραφο Α2.στ).

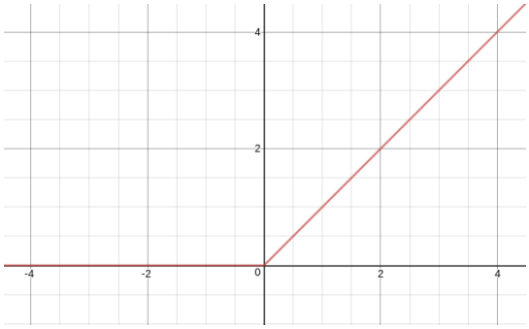
Για τη συνάρτηση ενεργοποίησης υπάρχουν επιλογές, όπως (εξετάζονται) η λογιστική σιγμοειδής συνάρτηση, η υπερβολική εφαπτομένη, η γραμμική, η Relu και άλλες οι οποίες στα πλαίσια της εργασίας δεν αναλύονται και συνεπώς δεν εξετάζονται. Κριτήρια επιλογής συνάρτησης ενεργοποίησης μπορούν να αποτελέσουν το είδος και η μορφή των δεδομένων, καθώς και το σφάλμα και η ταχύτητα εκπαίδευσης του δικτύου. Η επιλογή της συνάρτησης ενεργοποίησης πραγματοποιήθηκε μέσα από πειραματική διαδικασία με βάση τα παραπάνω κριτήρια.

Ως συνάρτηση ενεργοποίησης για το κρυφό επίπεδο κόμβων επελέγη η 'Relu'[7].

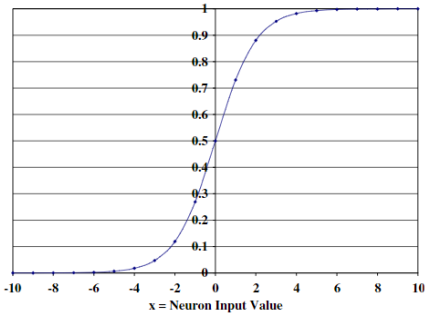
$$\boldsymbol{max(0,x)}, \qquad \boldsymbol{f(x)} = \begin{cases} \boldsymbol{0}, & \boldsymbol{x < 0} \\ \boldsymbol{x}, & \boldsymbol{x \geq 0} \end{cases}$$

[6] K. Potdar, T. S., and C. D., "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers," IJCA, vol. 175, no. 4, pp. 7–9, Oct. 2017, doi: 10.5120/ijca2017915495.
(https://www.researchgate.net/profile/Kedar_Potdar/publication/320465713_A_Comparative_Study_of_Categorical_Variable_Encoding_Techniques_for_Neural_Network_Classifiers/links/59e6f9554585151e5465859c/A-Comparative-Study-of-Categorical-Variable-Encoding-Techniques-for-Neural-Network-Classifiers.pdf)

[7] A. F. Agarap, "Deep Learning using Rectified Linear Units (ReLU)," *arXiv:1803.08375 [cs, stat]*, Feb. 2019.
(https://arxiv.org/pdf/1803.08375.pdf)



Εικόνα 1 The Rectified Linear Unit (ReLU) activation function[7]



Εικόνα 2 Sigmoid activation function [8]

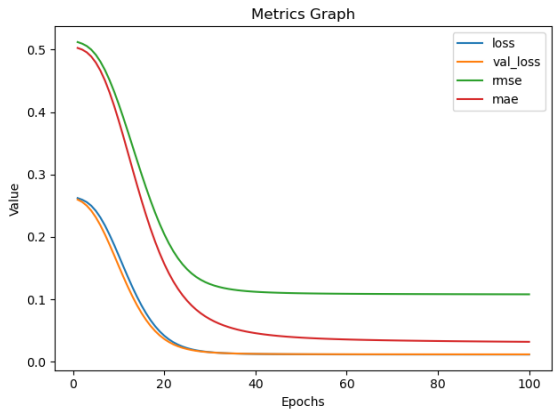
Στην επόμενη παράγραφο (Α2.ε) αναλύονται περισσότερο οι λόγοι επιλογής της συγκεκριμένης συνάρτησης, ως συνάρτηση ενεργοποίησης για το επίπεδο κρυφών κόμβων.

ε. Συνάρτηση ενεργοποίησης επιπέδου εξόδου

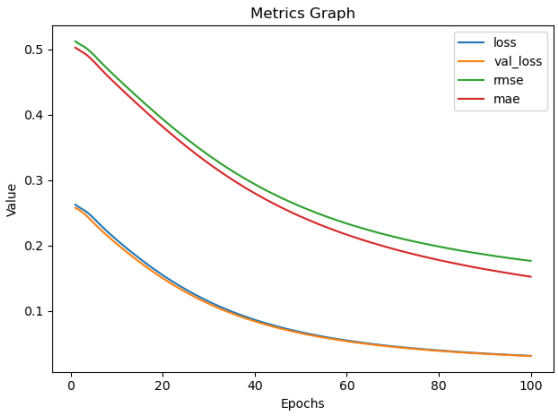
Ως συνάρτηση ενεργοποίησης για το επίπεδο εξόδου, αλλά και για το επίπεδο εισόδου του νευρωνικού δικτύου επελέγη **η σιγμοειδής συνάρτηση**[8]. Για την επιλογή αυτή ελήφθη υπ' όψιν ότι τα δεδομένα μετά το κεντράρισμα και την κανονικοποίηση βρίσκονται στο εύρος [-1,1] και όχι στο [0,1], αλλά και ότι είναι συνάρτηση παραγωγίσιμη, δίνοντας ταυτόχρονα πολύ καλά αποτελέσματα.

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

Τα παραπάνω επιβεβαιώνονται και από τα πειράματα στην python, εκ των οποίων ενδεικτικά γραφήματα παρουσιάζονται:

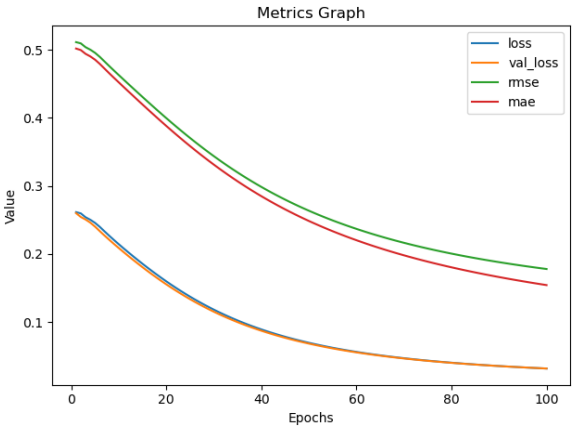


Εικόνα 3 inner level: relu, exit level: sigmoid

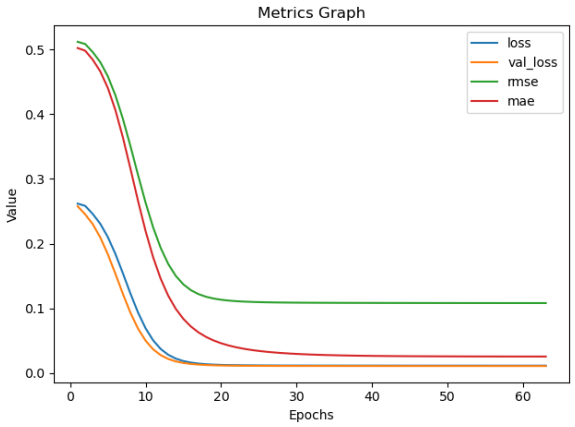


Εικόνα 4 inner level: sigmoid, exit level: sigmoid

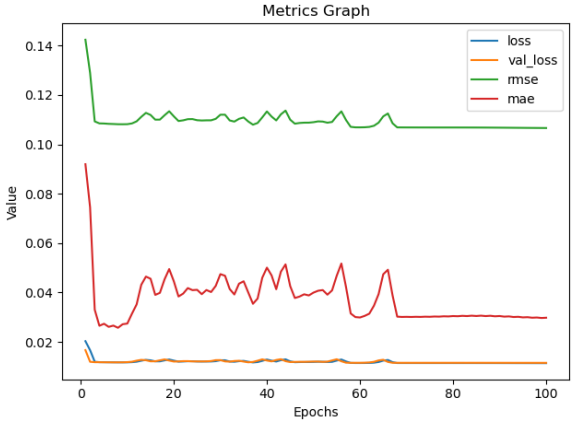
[8] J. F. Kros, M. Lin, and M. L. Brown, "Effects of the neural network s-Sigmoid function on KDD in the presence of imprecise data," Computers & Operations Research, vol. 33, no. 11, pp. 3136–3149, Nov. 2006, (https://reader.elsevier.com/reader/sd/pii/S0305054805000250?token=A5FAAE3E660A1308CD3FED8329576A1460CB19223CF40908E022D308D545ACD14D518ADE54D8C1D203EFEA4694D7BA7D)



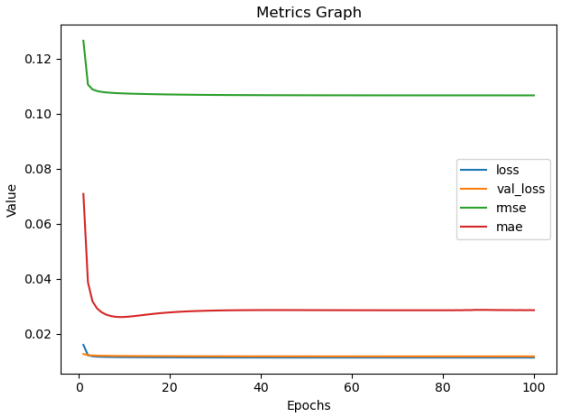
Εικόνα 5 inner level: tanh, exit level: sigmoid



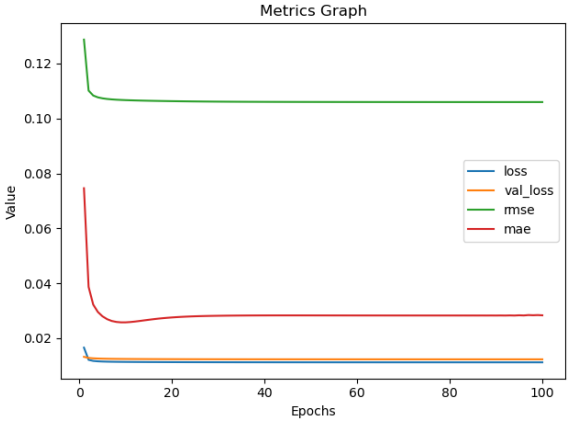
Εικόνα 6 inner level: linear, exit level: sigmoid



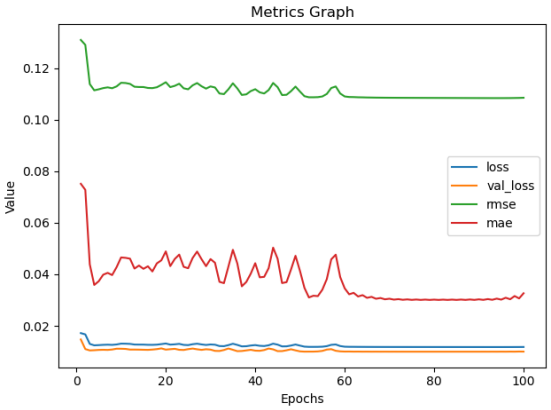
Εικόνα 7 inner level: linear, exit level: linear



Εικόνα 8 inner level: sigmoid, exit level: linear



Εικόνα 9 inner level: sigmoid, exit level: tanh



Εικόνα 10 inner level: tanh, exit level: tanh

στ. Αριθμός νευρώνων κρυφού επιπέδου

Γνωρίζουμε ότι στο αρχικό σύνολο δεδομένων υπάρχει απουσία χαρακτηριστικών (features) για τους χρήστες και τις ταινίες· οι χρήστες και οι ταινίες αναπαρίστανται από λανθάνοντα διανύσματα χαρακτηριστικών που καλούνται ενσωματώσεις (embeddings), τα οποία δεν είναι ακόμα γνωστά και που το ΤΝΔ καλείται να «μάθει». Ουσιαστικά, οι τιμές των διανυσμάτων θα αποθηκευτούν στα βάρη του δικτύου. Η ζητούμενη αξιολόγηση θα είναι το εσωτερικό γινόμενο των δύο διανυσμάτων χρήστη-ταινίας.

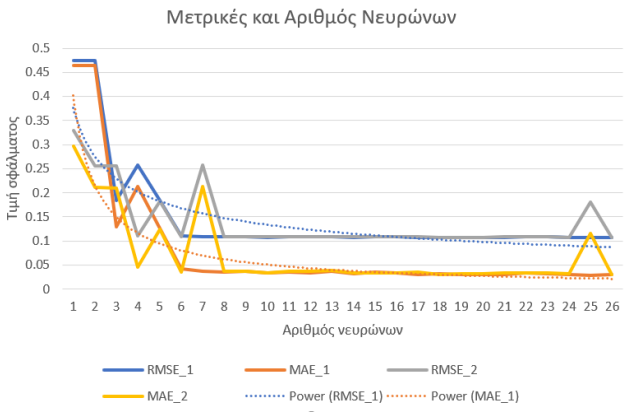
Υπενθύμιση: N: αριθμός χρηστών, M: αριθμός ταινιών

Έστω K ο αριθμός των κόμβων του κρυφού επιπέδου.
Έστω U πίνακας διάστασης $N \times K$ που προκύπτει από τις ενσωματώσεις των χρηστών.
Έστω F πίνακας διάστασης $K \times M$ που προκύπτει από τις ενσωματώσεις των ταινιών.

Θα πρέπει το γινόμενο UF να είναι ο πίνακας των αξιολογήσεων όλων των χρηστών για όλες τις ταινίες. Με τον τρόπο αυτό επιτυγχάνεται μείωση της διασταστικότητας, αφού αντί για έναν πίνακα $N \times M$, το ΤΝΔ καλείται να μάθει δύο πίνακες μικρότερης διάστασης. Οι τιμές των ζητούμενων πινάκων θα αποθηκεύονται στα βάρη του δικτύου.

Αρ. Νευρ.	RMSE_1	MAE_1	RMSE_2	MAE_2
1	0.4745	0.4649	0.32858	0.29677
2	0.47428	0.4647	0.25532	0.21107
3	0.18365	0.1288	0.25631	0.20975
4	0.25742	0.2132	0.11113	0.04489
5	0.18322	0.126	0.18207	0.12404
6	0.11035	0.0418	0.10852	0.03548
7	0.10878	0.0366	0.25668	0.21303
8	0.10863	0.0351	0.10881	0.0365
9	0.10869	0.0364	0.10914	0.03727
10	0.1079	0.0328	0.10824	0.03347
11	0.10882	0.0357	0.10894	0.03656
12	0.1082	0.0339	0.10896	0.0364
13	0.10894	0.0367	0.10964	0.03882
14	0.10784	0.0318	0.10842	0.03397
15	0.10872	0.0357	0.10827	0.03313
16	0.10813	0.0335	0.10832	0.03421
17	0.10787	0.0307	0.10861	0.03557
18	0.10792	0.032	0.10782	0.02981
19	0.10768	0.0308	0.10795	0.03199
20	0.10767	0.0305	0.10789	0.03131
21	0.10763	0.0294	0.10832	0.03365
22	0.10854	0.0334	0.10809	0.03309
23	0.1083	0.0327	0.10844	0.03286
24	0.10777	0.0306	0.10771	0.03108
25	0.10745	0.029	0.18004	0.11571
26	0.10761	0.0311	0.10781	0.03014
27	0.10769	0.0305	0.10759	0.02885
28	0.10765	0.0294	0.10744	0.0287
29	0.1074	0.0294	0.10741	0.02875
30	0.10743	0.0296	0.10775	0.02974

Από την παραπάνω σκέψη και ανάλυση προκύπτει το συμπέρασμα ότι $\boxed{K \ll N, M}$, δηλαδή $K \ll 943, 1682$. Προκειμένου, όμως να εντοπίσω το βέλτιστο, όσο γίνεται, αριθμό κόμβων κρυφού επιπέδου χρειάστηκε να κάνω δοκιμές για διάφορες (μικρές) τιμές του K. Δεν θα είχε νόημα, αν K > 30. Δοκιμές πραγματοποιήθηκαν (2 φορές) με πλήθος των νευρώνων κρυφού επιπέδου να κυμαίνεται από 1 μέχρι 30, με ρυθμό μάθησης η=0.001, ορμή m=0.2 και συναρτήσεις ενεργοποίησης relu για το κρυφό επίπεδο και σιγμοειδή για το επίπεδο εξόδου. Στο παρακάτω πίνακα και γράφημα φαίνονται τα αποτελέσματα των δοκιμών:



Εικόνα 11 Μετρικές και αριθμός νευρώνων κρυφού επιπέδου

Κάποιες χαρακτηριστικές τιμές από τα παραπάνω δεδομένα (από την πρώτη εκτέλεση), όπως και ζητείται, καταγράφονται στον παρακάτω πίνακα:

Αριθμός νευρώνων	RMSE	MAE
H = 4	0.25742	0.2132
H = 10	0.10790	0.0328
H = 20	0.10767	0.0305

Παρατηρείται ότι, με κάποιες εξαιρέσεις, για αριθμό νευρώνων από 8 και πάνω οι τιμές των μετρικών σταθεροποιούνται. Διατηρώντας, συνεπώς την αρχή να διατηρηθεί μικρή τιμή για το Κ, όπως και αναλύθηκε, για τα πειράματα επελέγει ο **αριθμός των νευρώνων κρυφού επιπέδου να είναι 10**· όπως φαίνεται και στο γράφημα της εικόνας 11, καθ' όλη την πορεία του γραφήματος υπάρχει μια πτωτική τάση στις τιμές των σφαλμάτων και επειδή κάθε πείραμα είναι ξεχωριστό, είναι ασφαλέστερη η επιλογή μεγαλύτερου αριθμού νευρωνών για το κρυφό επίπεδο από το 8. Δεν θα είχε νόημα για την παρούσα εργασία και το συγκεκριμένο σύνολο δεδομένων στο κρυφό επίπεδο να κρατάμε σχεδόν διπλάσιο όγκο πληροφορίας (στα βάρη) για να πετύχουμε ελαφρώς καλύτερο αποτέλεσμα και με ταυτόχρονο κίνδυνο να υπερεκπαιδευτεί πιο εύκολα τα δίκτυο. Για τους λόγους αυτούς επελέγει η τιμή 10.

ζ. Κριτήριο τερματισμού

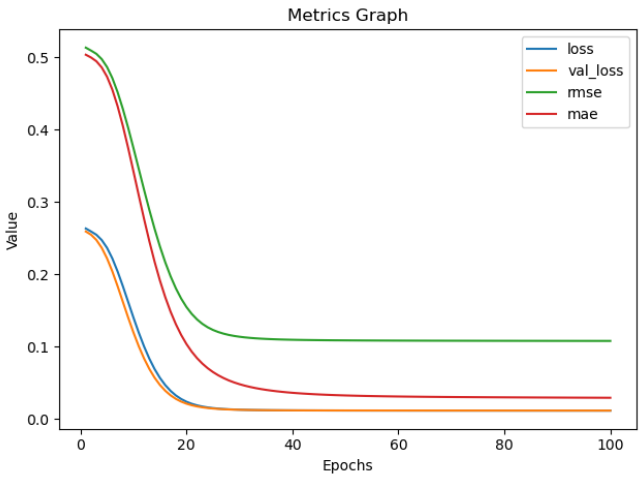
Βασικό στοιχείο τερματισμού της εκπαίδευσης είναι το πέρας συγκεκριμένου αριθμού εποχών. Κύριος στόχος κατά την εκπαίδευση με καθένα από τα 5 fold είναι να επιτευχθεί το ελάχιστο δυνατό σφάλμα, ακόμα και 0. Υπάρχει, όμως, και ο κίνδυνος κατά της φάση της εκπαίδευσης το ΤΝΔ να υπερεκπαιδευτεί, με αποτέλεσμα μετά την εκπαίδευση να μην μπορεί να γενικεύει, δηλαδή να μπορεί να υποθέσει σωστές βαθμολογίες για κάποιο νέο χρήστη. Για να αποφευχθεί αυτό μπορεί να χρησιμοποιηθεί η τεχνική του του πρόωρου σταματήματος (early stopping).

Κριτήρια, συνεπώς, τερματισμού της εκπαίδευση του ΤΝΔ μπορούν να αποτελέσουν (ο έλεγχος πραγματοποιείται στο τέλος κάθε εποχής):

i. Η επίτευξη του μέγιστου αριθμού επαναλήψεων του αλγορίθμου, δηλαδή ο μέγιστος αριθμός εποχών
ii. Η υπέρβαση του μέγιστου χρονικού ορίου που έχει θέσει ο χρήστης
iii. Πρόωρο σταμάτημα με χρήση συνόλου επικύρωσης
iv. Πρόωρο σταμάτημα λόγω μικρής διαφοράς στην τιμή του διανύσματος βαρών μεταξύ δύο εποχών ή η σταθεροποίησή τους
v. Πρόωρο σταμάτημα λόγω μικρής διαφοράς στην τιμή του ολικού σφάλματος μεταξύ δύο εποχών ή μείωσής του κάτω από μια επιθυμητή τιμή ή η επίτευξη της βέλτιστης επίδοσης.

Τα παρακάτω αποτελέσματα προέρχονται από την εκπαίδευση του δικτύου με 10 κόμβους στο κρυφό επίπεδο, συνάρτηση ενεργοποίησης Relu για το κρυφό επίπεδο και σιγμοειδή για το επίπεδο εξόδου, ρυθμό μάθησης η=0.001 και ορμή m=0.2, ενώ το μοντέλο εκπαιδεύτηκε για 100 εποχές, χωρίς early stopping .

Epoch	RMSE	MAE	loss
1	0.51309	0.50331	0.26326
2	0.50901	0.49944	0.25909
3	0.50455	0.49426	0.25457
4	0.49732	0.48582	0.24732
5	0.48639	0.47305	0.23658
6	0.47117	0.45522	0.22202
⋮	⋮	⋮	⋮
17	0.19724	0.14830	0.03890
18	0.18058	0.13101	0.03261
19	0.16648	0.11623	0.02771
20	0.15477	0.10367	0.02395
21	0.14524	0.09307	0.02110
22	0.13760	0.08413	0.01893
23	0.13156	0.07662	0.01731
24	0.12682	0.07030	0.01608
25	0.12312	0.06497	0.01516
26	0.12024	0.06047	0.01446
27	0.11801	0.05666	0.01392
28	0.11621	0.05335	0.01350
29	0.11483	0.05059	0.01318
30	0.11373	0.04821	0.01293
31	0.11286	0.04617	0.01274
32	0.11215	0.04440	0.01258
33	0.11158	0.04286	0.01245
34	0.11111	0.04152	0.01235
35	0.11073	0.04035	0.01226
36	0.11041	0.03933	0.01219
37	0.11014	0.03842	0.01213
38	0.10991	0.03762	0.01208
39	0.10972	0.03692	0.01204
40	0.01200	0.10955	0.03628
41	0.01197	0.10941	0.03572
42	0.01194	0.10928	0.03522
43	0.01192	0.10918	0.03477
44	0.01191	0.10908	0.03436
⋮	⋮	⋮	⋮
98	0.10796	0.02947	0.01165
99	0.10795	0.02945	0.01165
100	0.10794	0.02942	0.01165



Εικόνα 12 Γράφημα για εξέταση εποχών

Από τα στοιχεία του πίνακα:

Μεταβολή σφάλματος (Δe)	Μετρική	Epoch
Δe < 1%	RMSE	22 - 23
	MAE	23 - 24
Δe < 0.1%	RMSE	31 - 32
	MAE	40 - 41

Φαίνεται ότι 41 εποχές είναι αρκετές για την εκπαίδευση του μοντέλου. Επειδή, όμως, σε κάθε πείραμα διαφοροποιούνται τιμές, όπως ο ρυθμός μάθησης και η ορμή, ως αριθμός εποχών για την εκπαίδευση επελέγη **epochs = 50**.

Προκειμένου να μην υπερεκπαιδευτεί το νευρωνικό δίκτυο, ειδικά τώρα που η τιμή 50 των εποχώ είναι μικρότερη του 41 που βρέθηκε ως μια βέλτιστη τιμή στο προηγούμενο πείραμα, χρησιμοποιήθηκε και η τεχνική του πρόωρου σταματήματος. Στα πλαίσια αυτής της εργασίας από τα κριτήρια τερματισμού που αναφέρθηκαν και αφορούν το πρόωρο σταμάτημα (κριτήρια iii-v) εξετάστηκε μόνο η τελευταία περίπτωση, δηλαδή **πρόωρο σταμάτημα λόγω μικρής διαφοράς στην τιμή του ολικού σφάλματος μεταξύ δύο εποχών ή μείωσής του κάτω από μια επιθυμητή τιμή ή η επίτευξη της βέλτιστης επίδοσης**. Αυτό στην python υλοποιήθηκε με τη συνάρτηση `EarlyStopping()`[9] και με βάση τη τιμή του MAE, όταν δεν υπάρχει μεταβολή πάνω από 0.05%.

[9] https://keras.io/callbacks/

η. Άλλες παράμετροι (loss, optimizer, batch_size, epochs)

Κατά τη δημιουργία και την εκπαίδευση του νευρωνικού δικτύου με Python χρειάζεται και ο καθορισμός κάποιων επιπλέον παραμέτρων, όπως στην εντολή compile(): loss, optimizer και στην fit(): batch_size.

loss

Για την παράμετρο loss της συνάρτησης model.compile() χρησιμοποιήθηκε η **mean_squared_error.**

optimizer

Για την παράμετρο optimizer της συνάρτησης model.compile() χρησιμοποιήθηκε **nadam**[10]**.**
Optimazer του μοντέλου έχει οριστεί **SGD.**

batch_size

Αφορά το μέγεθος των δειγμάτων σε κάθε ενημέρωση του μοντέλου κατά την εκπαίδευσή του. Ως τιμή του *batch_size* ορίστηκε το μέγεθος του διανύσματος εισόδου, το οποίο έχει καθοριστεί να είναι **N = 943** (βλέπε υποερώτημα Α2.β). Για την παράμετρο batch_size της συνάρτησης model.fit() χρησιμοποιήθηκε το **μέγεθος του διανύσματος εισόδου**, δηλαδή έμμεσα το μέγεθος 943.

[10] T. Dozat, "Incorporating Nesterov Momentum into Adam," p. 6. (http://cs229.stanford.edu/proj2015/054_report.pdf)

Α3. Μεταβολές στο ρυθμό εκπαίδευσης και σταθεράς ορμής

Με βάση τις αναλύσεις που προηγήθηκαν, για τα παρακάτω πειράματα χρησιμοποιήθηκαν οι εξής παράμετροι:

kfold cross validation	***K = 5***
Μέγεθος διανύσματος εισόδου	***N = 943***
Μέγεθος διανύσματος εξόδου	***M = 1682***
Αριθμό νευρώνων του κρυφού επιπέδου	***10***
Συνάρτηση ενεργοποίησης κρυφού επιπέδου	***relu***
Συνάρτηση ενεργοποίησης επιπέδου εξόδου	***sigmoid***
Εποχές εκπαίδευσης	***50***
Early stopping	***Με βάση το MAE (Δe<0.05%)***
Μετρικές σφάλματος	***RMSE, MAE***
Batch_size	***N = 943***
loss	***mean_squared_error***

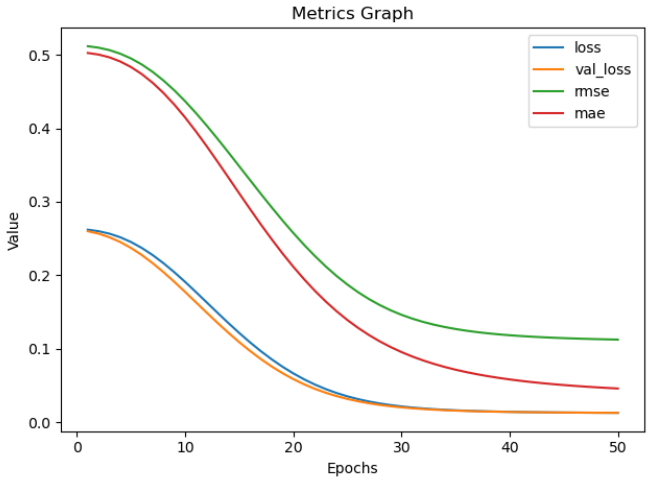
Σε αυτό το ερώτημα πραγματοποιείται βελτιστοποίηση των υπερπαραμέτρων ρυθμού εκπαίδευσης **η** και σταθεράς ορμής **m** με χρήση Cross Validation. Τα συνολικά αποτελέσματα, τα οποία αναλύονται παρακάτω, παρουσιάζονται στον πίνακα:

η	**m**	**RMSE**	**MAE**	**[Εποχές]**
0.001	0.2	0.11286	0.04708	48
0.001	0.6	0.11158	0.04357	48
0.05	0.6	0.11147	0.04241	46
0.1	0.6	0.11053	0.04187	47

Παρατηρήσεις:

1. *Λόγω χρήσης του 5-kfold και του early stopping, τα γραφήματα αναπαριστούν τις γραφικές παραστάσεις σφαλμάτων μόνο της εκτέλεσης του αλγορίθμου με το τελευταίο fold.*
2. [Εποχές] → άνω όριο του μέσου όρου των εποχών (λόγω του 5-fold)

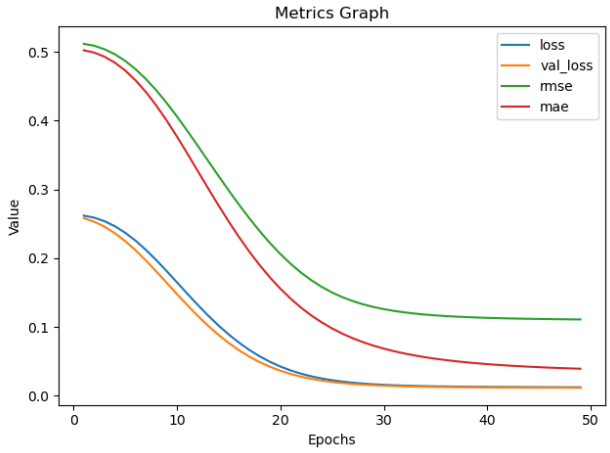
Για η = 0.001 και m = 0.2



fold	epochs	RMSE	MAE
1	50	0.12027	0.06429
2	50	0.11283	0.06751
3	50	0.11493	0.04613
4	40	0.10463	0.03229
5	50	0.11162	0.04643

Εικόνα 13 Μετρικές σφαλμάτων για η=0.001 και m=0.2

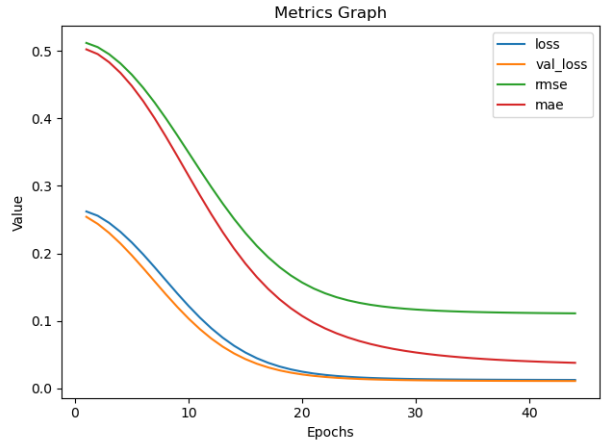
Για η = 0.001 και m = 0.6



fold	epochs	RMSE	MAE
1	49	0.10476	0.03719
2	50	0.12052	0.06563
3	49	0.10717	0.03807
4	43	0.11870	0.03851
5	49	0.10675	0.03846

Εικόνα 14 Μετρικές σφαλμάτων για η=0.001 και m=0.6

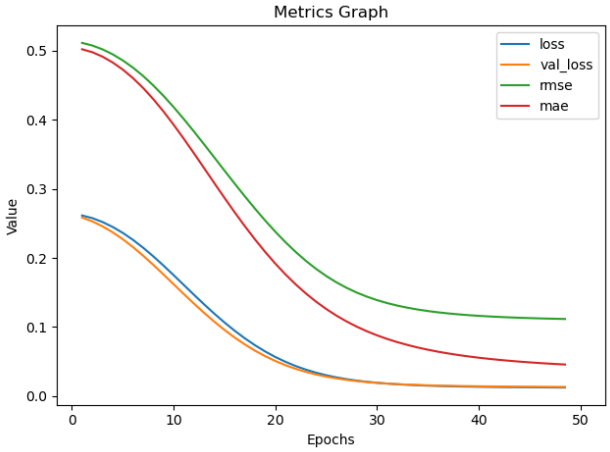
Για η = 0.05 και m = 0.6



fold	epochs	RMSE	MAE
1	50	0.11554	0.06206
2	45	0.11076	0.03798
3	45	0.11193	0.03801
4	44	0.11556	0.03872
5	44	0.10355	0.03527

Εικόνα 15 Μετρικές σφαλμάτων για η = 0.05 και m = 0.6

Για η = 0.1 και m = 0.6



fold	epochs	RMSE	MAE
1	50	0.10928	0.04395
2	46	0.10882	0.03896
3	41	0.10694	0.03455
4	50	0.11824	0.04830
5	48	0.10730	0.03779

Εικόνα 16 Μετρικές σφαλμάτων για η = 0.1 και m = 0.6

Είναι γνωστό ότι όσο πιο μικρή είναι η παράμετρος μάθησης ***η*** τόσο πιο μικρές θα είναι οι αλλαγές στα βάρη του δικτύου σε κάθε επανάληψη του συνόλου εκπαίδευσης και τόσο πιο ομαλή θα είναι η φθίνουσα καμπύλη στο υπερεπίπεδο. Αν όμως η παράμετρος μάθησης γίνει πολύ μεγάλη, έτσι ώστε να επιταχύνει τον ρυθμό μάθησης, οι μεγάλες αλλαγές που θα υπάρξουν στα βάρη μπορεί να δημιουργήσουν ένα ασταθές δίκτυο. Για το λόγω αυτό χρησιμοποιούμε κατά την εκπαίδευση και την ορμή, ώστε ταυτόχρονα και να αυξήσουμε το ρυθμό μάθησης και να αποφύγουμε τον κίνδυνο της αστάθειας. Έτσι έχουμε:

Ισχύει από τον κανόνα δέλτα[11]: $\Delta w_{ij}(n) = -\eta \frac{\partial E(n)}{\partial w_{ij}(n)}$

Αν ορίσουμε α τη σταθερά της ορμής (momentum constant) που συνήθως είναι ένας θετικός αριθμός και καθορίζει πόσο μεγάλη είναι η αλλαγή του βάρους στον επόμενο υπολογισμό, τότε προκύπτει:

$$\Delta w_{ij}(n) = a\Delta w_{ij}(n-1) + \eta\delta_j(n)y_j(n) \Rightarrow$$
$$\Rightarrow \Delta w_{ij}(n) = -\eta \sum_{t=0}^{n} a^{n-\tau}\delta_j(n)y_j(n) \Rightarrow$$
$$\xrightarrow{\delta_j(n)y_j(n)=\frac{\partial E(n)}{\partial w_{ij}(n)}} = -\eta \sum_{t=0}^{n} a^{n-\tau}\frac{\partial E(n)}{\partial w_{ij}(n)}$$

Από την τελευταία σχέση φαίνεται πως ο παράγοντας διόρθωσης βάρους $\Delta w_{ij}(n)$ αποτελείται από άθροισμα εκθετικών όρων. Για να συγκλίνει αυτή η σειρά πρέπει υποχρεωτικά $\boxed{0 < |\alpha| < 1}$.

Όπως φαίνεται και από τα παραπάνω πειράματα χρησιμοποιήθηκαν τιμές για την ορμή μικρότερες της μονάδας. Επίσης, από τα πειράματα αποδεικνύεται ότι όσο αυξάνεται ο ρυθμός μάθησης ***η*** ή η ορμή ***m***, τόσο πιο γρήγορα συγκλείνει ο αλγόριθμός και έχουμε πιο γρήγορα αποτελέσματα. Αυτό συμβαίνει στα πειράματα με τιμές [η, m]=[0.001, 0.2], [η, m]=[0.001, 0.6], [η, m]=[0.05, 0.6], όπως φαίνεται τόσο από τα διαγράμματα όπου η κατάβαση των καμπυλών γίνεται πιο απότομη, όσο και από τους πίνακες. Όταν, όμως, αυξήθηκαν ακόμα περισσότερο οι τιμές [η, m]=[0.001, 0.2], μπορεί ο αλγόριθμος να ολοκληρωνόταν σε περισσότερες εποχές, αλλά με το πέρας του, οι τιμές των RMSE και MAE εξακολουθούσαν ένα είναι ελαφρώς χαμηλότερες από τις υπόλοιπες περιπτώσεις. Ως καλύτερος συνδυασμός των παραμέτρων ρυθμού μάθησης και ορμής, από τους παραπάνω που εξετάστηκαν, θα μπορούσε να επιλεγεί: [η, m]=[0.05, 0.6]. Συνεπώς, μπορούμε μέχρι ενός σημείου να αυξήσουμε το ρυθμό μάθησης και της ορμής, αρκεί να μην ξεπεράσουμε ένα όριο και έχουμε αντίθετα από τα επιθυμητά αποτελέσματα.

[11] Σ. Λυκοθανάσης, Ε. Γεωργόπουλος, "ΥΠΟΛΟΓΙΣΤΙΚΗ ΝΟΗΜΟΣΥΝΗ, ΠΑΝΕΠΙΣΤΗΜΙΑΚΕΣ ΣΗΜΕΙΩΣΕΙΣ", Πανεπιστήμιο Πατρών, Πάτρα 2014

Α4. Ομαλοποίηση

Μια μέθοδος για την αποφυγή υπερπροσαρμογής του δικτύου και βελτίωση της γενικευτικής του ικανότητας είναι η ομαλοποίηση του διανύσματος των βαρών (regularization). Για το λόγω αυτό, όπως και ζητείται, εφάρμοσα L1 ομαλοποίηση και επανεκπαίδευσα το ΤΝΔ, όπως προέκυψε από το Α3 ([η, m]=[0.05, 0.6]), για συντελεστή φθοράς r = 0.1, r = 0.5 και r = 0.9.

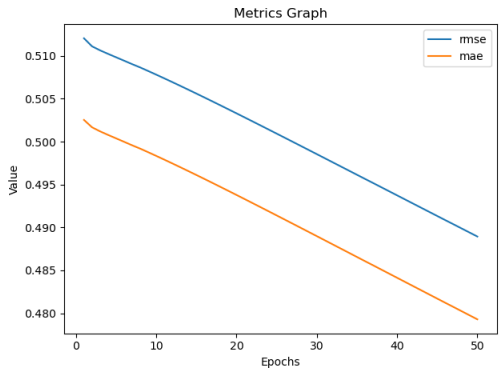
Στη συγκεκριμένη περίπτωση είναι προτιμότερη η χρήση L1 ομαλοποίησης έναντι L2. Πιο συγκεκριμένα, η L1 ομαλοποίηση, χρησιμοποιεί έναν όρο ποινής που ενθαρρύνει το άθροισμα των απόλυτων τιμών των παραμέτρων να είναι μικρό, ενώ η L2 ομαλοποίηση ενθαρρύνει το άθροισμα των τετραγώνων των παραμέτρων να είναι μικρό. Έχει συχνά παρατηρηθεί ότι η L1 ομαλοποίηση σε πολλά μοντέλα μηδενίζει πολλές παραμέτρους, με αποτέλεσμα το διάνυσμα παραμέτρων να είναι αραιό. Αυτό το καθιστά ένα φυσικό υποψήφιο στις ρυθμίσεις επιλογής χαρακτηριστικών, όπου πιστεύουμε ότι πολλές λειτουργίες πρέπει να αγνοηθούν[12]. Η L1 ομαλοποίηση, συνεπώς, σε σύγκριση με την L2 μπορεί πιο εύκολα να μηδενίσει την τιμή κάποιου κόμβου («άχρηστος κόμβος»), μειώνοντας τις τιμές στα βάρη του, για αυτό και στα πλαίσια της συγκεκριμένης εργασία με το δοθέν σύνολο δεδομένων προτιμάται.

Παρατηρήθηκε ότι, όταν σαν παράμετρος μπήκε και η ομαλοποίηση, αρχικά λόγο του πρόωρου σταματήματος, κάθε fold έτρεχε για μόλις 2 εποχές. Αφαιρώντας από το πρόωρο σταμάτημα την παράμετρο διαφοράς MAE μικρότερη από 0.05% σε δύο συνεχόμενες εποχές, αλλά διατηρώντας το κριτήριο min, κάθε fold έτρεξε το μέγιστο αριθμό εποχών, ενώ από το γράφημα που προέκυψε, φάνηκε η συνεχής πτωτική τάση, αλλά με πολύ μικρή κλίση, που έχουν οι μετρικές των σφαλμάτων.

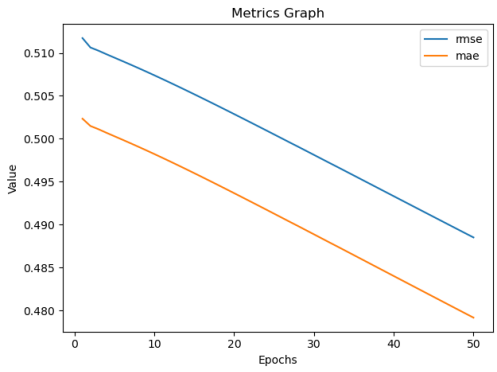
Οι ζητούμενες τιμές συμπληρώθηκαν με αριθμό εποχών 50 και ως κριτήριο πρόωρου σταματήματος η ελάχιστη τιμή της μετρικής MAE (χωρίς το επιπλέον κριτήριο της διαφοράς ανά εποχή).

Τα συνολικά αποτελέσματα από τα πειράματα για τις διάφορες τιμές ομαλοποίησης που ζητήθηκαν (r1-r3), τα οποία αναλύονται παρακάτω, παρουσιάζονται στον πίνακα:

Συντελεστής φθοράς	RMSE	MAE
0.1	0.48892	0.47947
0.5	0.48884	0.47939
0.9	0.48874	0.47929

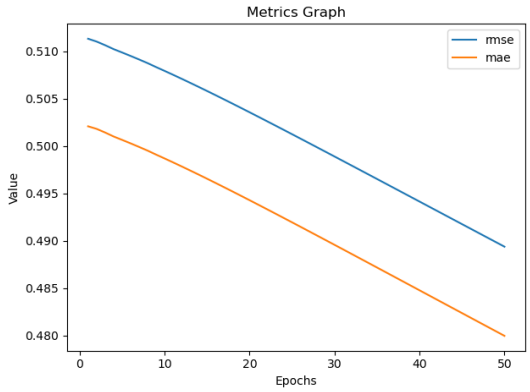


Εικόνα 17 Εκπαίδευση με r=0.1

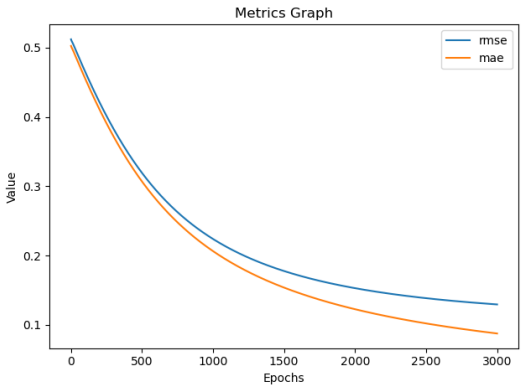


Εικόνα 18 Εκπαίδευση με r=0.5

[12] A. Y. Ng, "Feature selection, L 1 vs. L 2 regularization, and rotational invariance," in Twenty-first international conference on Machine learning - ICML '04, Banff, Alberta, Canada, 2004, p. 78, (https://dl.acm.org/doi/abs/10.1145/1015330.1015435)



Εικόνα 19 Εκπαίδευση με r=0.9



Εικόνα 20 Εκπαίδευση με r=0.5 για 3.000 εποχές

Δοκιμάστηκε ο συγκεκριμένος αλγόριθμος με συντελεστή ομαλοποίησης r = 0.5 και για 3.000 εποχές. Τα αποτελέσματα φαίνονται στην εικόνα 20 και οι τιμές των μετρικών στο διπλανό πίνακα.

fold	RMSE	MAE
1	0.12759	0.08682
2	0.13085	0.08815
3	0.13033	0.08760
4	0.12827	0.08735
5	0.12474	0.08526
mean	**0.12835**	**0.08704**

Ένα πρώτο συμπέρασμα που προκύπτει είναι ότι με τον παράγοντα της ομαλοποίησης ο αλγόριθμος χρειάζεται πολλές περισσότερες εποχές για να εκπαιδευτεί με το ελάχιστο δυνατό σφάλμα, όπως αυτό προκύπτει από τις μετρικές RMSE και MAE, αλλά και από το γράφημα της εικόνας 20. Αυτό μπορεί να οφείλεται στο γεγονός ότι, λόγω της ομαλοποίησης μηδενίζονται και χρήσιμοι κόμβοι, με αποτέλεσμα το νευρωνικό δίκτυο να χρειάζεται περισσότερο χρόνο προκειμένου να εκπαιδευτεί με τους εναπομείναντες.

Ένα δεύτερο συμπέρασμα το οποίο προκύπτει τόσο από τον πίνακα αποτελεσμάτων με τις τιμές των μετρικών για τις διάφορες τιμές του παράγοντα ομολοποίησης που δοκιμάστηκαν, όσο και από την κλίση των γραφικών παραστάσεων των μετρικών, είναι ότι όσο αυξάνεται η τιμή του r, τόσο πιο γρήγορα φαίνεται να συγκλίνει ο αλγόριθμός, καθώς οι τιμές των μετρικών σφαλμάτων δείχνουν να μειώνονται.

Ως προς τη γενικευτική ικανότητα του δικτύου, καθώς ο αλγόριθμος τερματίζει με υψηλότερες τιμές στις μετρικές RMSE και MAE, χωρίς να έχει υπερεκπαιδευτεί, (κρατώντας ίδιες όλες τις υπόλοιπες παραμέτρους με τα πειράματα των προηγούμενων ερωτημάτων) γεγονός που προκύπτει από τα πρώτα 2 συμπεράσματα, το νευρωνικό δίκτυο, χρησιμοποιώντας τον παράγοντα ομαλοποίησης, μπορεί να γενικεύει καλύτερα, καθώς δεν εξειδικεύεται πλήρως στο συγκεκριμένο σύνολο δεδομένων. Έτσι, για κάθε νέο δείγμα-χρήστη, θα μπορεί πιο εύκολα και πιθανότατα με μεγαλύτερη επιτυχία, να προβλέψει τις βαθμολογίες του.

Α5. Βαθύ Νευρωνικό Δίκτυο

Δοκιμάστηκε να προστεθούν ακόμη μέχρι και 1 επιπλέον κρυφά επίπεδα (τελικός συνολικός αριθμός κρυφών επιπέδων: 2, συνολικός αριθμός επιπέδων 4). Για το συγκεκριμένο ερώτημα δεν χρησιμοποιήθηκε ο παράγοντας της ομαλοποίησης, λόγω μεγάλης χρονικής καθυστέρησης. Καθένα από τα επιπλέον επίπεδα εξετάστηκε ως προς τον αριθμό των κόμβων του σε σχέση με το αποτέλεσμα που παρείχαν στην έξοδο του δικτύου. Ως συνάρτηση ενεργοποίησης των κρυφών επιπέδων επελέγει η **Relu**.

Αρχικά, με βάση τα προηγούμενα ερωτήματα χρησιμοποιήθηκαν οι παρακάτω τιμές:

kfold cross validation	***K = 5***
Μέγεθος διανύσματος εισόδου	***N = 943***
Μέγεθος διανύσματος εξόδου	***M = 1682***
Αριθμό νευρώνων του κρυφού επιπέδου	***10***
Συνάρτηση ενεργοποίησης κρυφών επιπέδων	***relu***
Συνάρτηση ενεργοποίησης επιπέδου εξόδου	***sigmoid***
Εποχές εκπαίδευσης	***50***
Early stopping	***Με βάση το MAE (Δe<0.05%)***
Μετρικές σφάλματος	***RMSE, MAE***
Batch_size	***N = 943***
loss	***mean_squared_error***
activity_regularizer	***0***

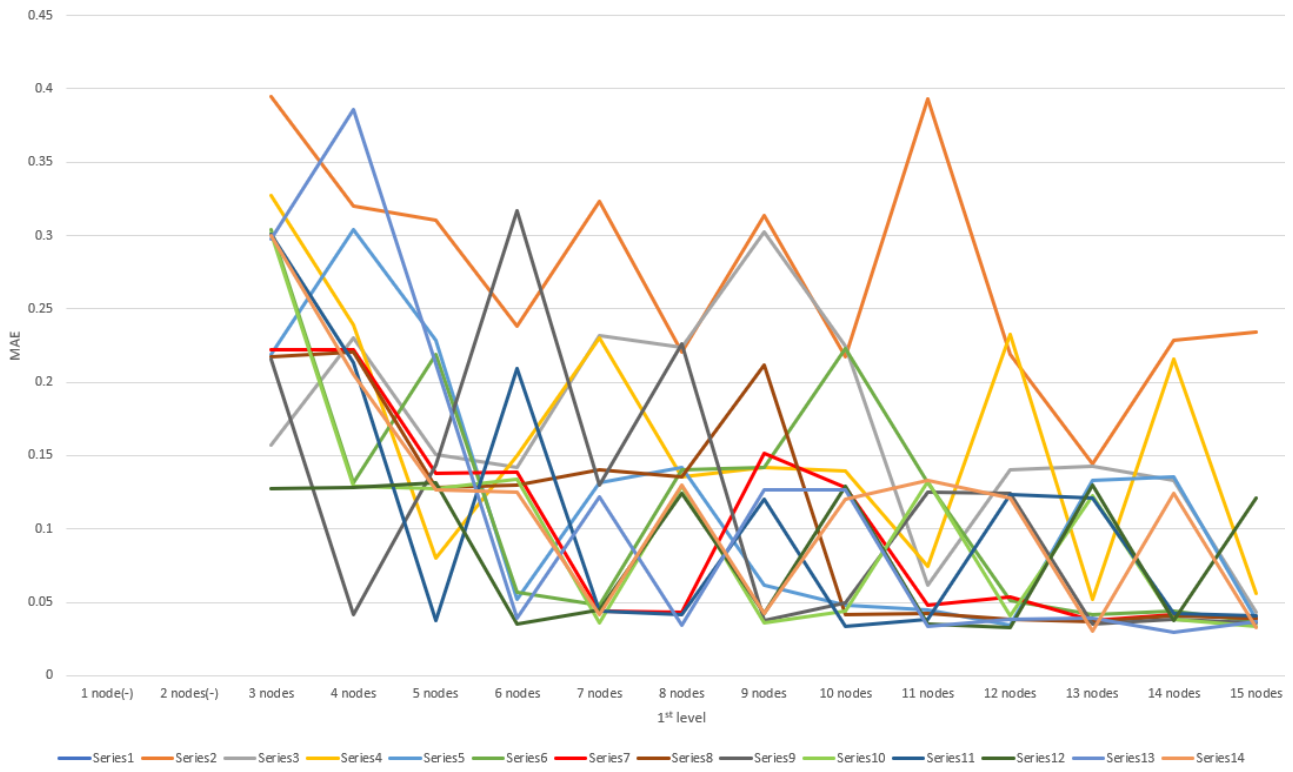
Στη συνέχεια, προκειμένου να βρεθεί ο βέλτιστος συνδυασμός αριθμού κόμβων των 2 κρυφών επιπέδων, εξετάστηκαν όλοι οι πιθανοί συνδυασμοί σε συγκεκριμένο εύρος. Από την υποενότητα Α2.στ φάνηκε ότι δεν έχει νόημα να χρησιμοποιηθούν λιγότεροι από 4 κόμβοι και περισσότεροι από 10. Για το λόγο αυτό, θεωρήθηκε ότι το πρώτο κρυφό επίπεδο μπορεί να έχει από 3 μέχρι 15 [3, 15] κόμβους, ενώ το δεύτερο από 2 μέχρι 14 [2, 14] κόμβους. Στον πίνακα που ακολουθεί παρουσιάζονται τα αποτελέσματα από τους συνδυασμούς, με βάση το κριτήριο MAE. Για καλύτερη κατανόηση των αποτελεσμάτων, παρέχεται και γραφική απεικόνιση.

2^{nd} \ 1^{st}	3 nodes	4 nodes	5 nodes	6 nodes	7 nodes	8 nodes	9 nodes	10 nodes	11 nodes	12 nodes	13 nodes	14 nodes	15 nodes
2	0.39469	0.32042	0.31046	0.23794	0.32299	0.22096	0.31363	0.21745	0.39343	0.21916	0.14449	0.22840	0.23430
3	0.15702	0.23018	0.15109	0.14226	0.23145	0.22385	0.30248	0.22425	0.06159	0.14067	0.14281	0.13278	0.04296
4	0.32750	0.23915	0.08048	0.14956	0.22997	0.13552	0.14161	0.13975	0.07461	0.23280	0.05211	0.21601	0.05589
5	0.21923	0.30439	0.22839	0.05166	0.13180	0.14224	0.06204	0.04791	0.04447	0.03469	0.13325	0.13513	0.03851
6	0.30397	0.13184	0.21919	0.05707	0.04807	0.14069	0.14155	0.22328	0.13161	0.05127	0.04165	0.04368	0.03669
7	0.22218	0.22190	0.13794	0.13892	0.04416	0.04331	0.15116	0.12819	0.04775	0.05323	0.03755	0.04165	0.03937
8	0.21708	0.22043	0.12791	0.12965	0.14040	0.13525	0.21159	0.04119	0.04266	0.03868	0.03686	0.04065	0.03922
9	0.21554	0.04128	0.14353	0.31683	0.13013	0.22585	0.03725	0.04980	0.12499	0.12405	0.03514	0.03807	0.03574
10	0.30141	0.12923	0.12772	0.13423	0.03557	0.12874	0.03613	0.04372	0.13252	0.04062	0.12258	0.03874	0.03365
11	0.30093	0.21348	0.03766	0.20943	0.04385	0.04143	0.12028	0.03360	0.03842	0.12312	0.12080	0.04219	0.04091
12	0.12716	0.12818	0.13186	0.03522	0.04496	0.12409	0.04218	0.12883	0.03542	0.03301	0.13017	0.03759	0.12133
13	0.29725	0.38591	0.21283	0.03916	0.12222	0.03417	0.12666	0.12652	0.03377	0.03816	0.03891	0.02954	0.03715
14	0.29982	0.20499	0.12661	0.12475	0.04185	0.13007	0.04206	0.12050	0.13292	0.12138	0.03065	0.12388	0.03299

Έχουν χρωματιστεί οι μικρότερες τιμές, που αποτελούν και κατώτατα στο γράφημα που έπεται.

Όπως φαίνεται και από το διάγραμμα (οι χαμηλότερες κορυφές), υπάρχουν αρκετοί πιθανοί συνδυασμοί που φαίνεται να λειτουργούν αποδοτικά· αυτοί οι συνδυασμοί έχουν καταγραφεί στο διπλανό πίνακα.

1ο Κρυφό επίπεδο	2ο Κρυφό επίπεδο	MAE
4	9	0.04128
5	11	0.03766
6	5	0.05166
6	12	0.03522
7	7	0.04416
7	10	0.03557
8	7	0.04331
8	11	0.04143
8	13	0.03417
9	9	0.03725
9	10	0.03613
10	8	0.04119
10	11	0.03360
11	11	0.03842
11	13	0.03377
12	12	0.03301
13	14	0.03065
14	13	0.02954
15	14	0.03299

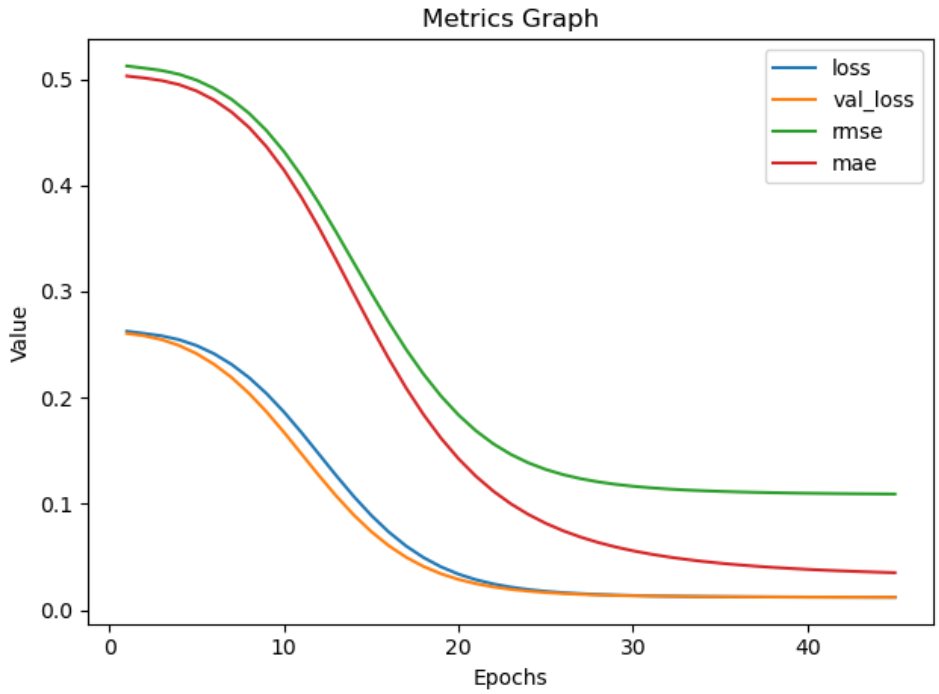


Εικόνα 21 Διακύμανση τιμής MAE ανάλογα με το συνδυασμό κόμβων των 2 κρυφών επιπέδων

**το κάθε χρώμα (Series X) του γραφήματος αντιστοιχεί σε συγκεκριμένο αριθμό κόμβου του δευτέρου κρυφού επιπέδου (το Series 1 δεν αποτυπώνεται στο γράφημα καθώς δεν υπάρχουν δεδομένα για αυτό) – Series 2: 2 nodes, Series 3: 3 nodes …*

Από τα αποτελέσματα, φαίνεται ότι με τη χρήση περισσότερων κόμβων το MAE έχει την τάση να μειώνεται. Επίσης, Υπάρχει η τάση οι μικρότερες τιμές της μετρικής MAE να βρίσκονται κοντά στη διαγώνιο του πίνακα, **δηλαδή όταν ο αριθμός κόμβων πρώτου και δεύτερου επιπέδου είναι ίσος**. Η μικρότερη τιμή MAE εντοπίστηκε στο συνδυασμό 14 και 13 κόμβων για το πρώτο και δεύτερο κρυφό επίπεδο αντίστοιχα.

Συνεπώς, με βάση αυτά τα κριτήρια και τα αποτελέσματα, επέλεξα για το δίκτυο να θέσω τον αριθμό κόμβων **του πρώτου κρυφού επιπέδου ίσο με 14 και του δεύτερο με 13**, δηλαδή τη βέλτιστη επιλογή από άποψη σφάλματος.



Εικόνα 22 Εκπαίδευση με 2 κρυφά επίπεδα, lr=0.05, m=0.6, 50 εποχές με early stopping

Πιο αναλυτικά τα αποτελέσματα:

fold	epochs	RMSE	MAE
1	38	0.10801	0.03027
2	38	0.11301	0.03244
3	38	0.10229	0.02990
4	45	0.10994	0.03521
5	45	0.10843	0.03528
mean	**41**	**0.10834**	**0.03262**
Με 1 κρυφό	**46**	0.11147	0.04241

Τα αποτελέσματα είναι ικανοποιητικά, καθώς εν συγκρίσει των αποτελεσμάτων με τις ίδιες τιμές παραμέτρων, αλλά ένα κρυφό επίπεδο (βλέπε ενότητα Α3), πετύχαμε με πρόωρο σταμάτημα σε λιγότερες εποχές καλύτερες τιμές για τις μετρικές σφαλμάτων RMSE και MAE. Συνεπώς, αν και για την εκπαίδευση του δικτύου, και από μαθηματική άποψη, αρκεί το ένα κρυφό επίπεδο, είναι φανερό πως με την προσθήκη ενός ακόμη κρυφού επιπέδου λαμβάνουμε καλύτερα αποτελέσματα και με καλύτερες επιδώσεις.

Υποχρεωτικός Κώδικας

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold
import keras
from keras.layers import Dense
from keras import backend as K
from keras.callbacks import EarlyStopping
from keras.regularizers import l1
import matplotlib.pyplot as plt

#####################################   parameters   #####################################
kfold_value = 5  # number of k for kfold CV
epochs_value = 50  # number of epochs
internal_neurons_number = 10  # number of neurons and internal nodes
min_MAE_distance = 0.0005  # early stopping when MAE(n)-MAE(n-1) < min_MAE_distance
learning_rate_value = 0.1  # value for lr parameter
momentum_value = 0.8  # value for momentum parameter
regularizer_value = 0  # value for regularizer

#####################################   Read dataset   #####################################
dataset = pd.read_csv('dataset/ml-100k/u.data', header=None, delimiter="\t", names=['userId', 'itemId', 'rating', 'timestamp'])
dataset = dataset.apply(pd.to_numeric)

#####################################   data_centering   #####################################
average_user_rating = {}
for userId in dataset.userId.unique():
    average_user_rating[userId] = dataset.loc[dataset['userId'] == userId, 'rating'].mean()  # centering
    dataset.loc[dataset['userId'] == userId, 'rating'] -= average_user_rating[userId]
print(max(dataset.loc[:, 'rating']))
print(min(dataset.loc[:, 'rating']))
#####################################   missing values   #####################################
# NxM_dataset: NxM Array with user ratings and 0 for missing values
NxM_dataset = pd.pivot_table(dataset, index='userId', columns='itemId', values='rating', fill_value=0)
NxM_dataset = NxM_dataset.apply(pd.to_numeric)

#####################################   data_normalization   #####################################
for userId in dataset.userId.unique():
    NxM_dataset.loc[userId, :] = NxM_dataset.loc[userId, :] / max(abs(NxM_dataset.loc[userId, :]))

```

```python
############################   5-fold_cross_validation   ############################
X = np.array(pd.get_dummies(dataset.userId.unique()))  # input vector
Y = np.array(NxM_dataset.loc[:, :])  #outputvector

kf = KFold(n_splits=kfold_value, shuffle=True)

#####################################   A-NN_model   #####################################
rmseList = []  # list for RSME metric
maeList = []  # list for MAE metric

fold = 0
for (train_index, test_index) in kf.split(X):
    fold += 1
    model = keras.Sequential()

    model.add(Dense(len(X), activation="sigmoid",  activity_regularizer=l1(regularizer_value), input_dim=len(X)))  # input level
    model.add(Dense(internal_neurons_number, activation="relu", activity_regularizer=l1(regularizer_value), input_dim=len(X)))  # internal level
    model.add(Dense(len(Y[0]), activation="sigmoid",  input_dim=internal_neurons_number))  # output level

    # rmse calculation function
    def rmse(y_true, y_pred):
        return K.sqrt(K.mean(K.square(y_pred - y_true)))

    # mae calculation function
    def mae(y_true, y_pred):
        return K.mean(K.abs(y_pred - y_true))

    keras.optimizers.SGD(lr=learning_rate_value, momentum=momentum_value, decay=0.0, nesterov=False)
    model.compile(loss='mean_squared_error', optimizer='nadam', metrics=[rmse, mae])

    early_stopping = EarlyStopping(monitor='mae', mode='min', min_delta=min_MAE_distance, verbose=1)
    history = model.fit(X[train_index], Y[train_index], validation_data=(X[test_index], Y[test_index]), epochs=epochs_value, batch_size=len(X), callbacks=[early_stopping], verbose=0)

    scores = model.evaluate(X[test_index], Y[test_index], verbose=0)
    rmseList.append(scores[1])
    maeList.append(scores[2])

    print("Fold :", fold, " RMSE:", scores[1], " MAE:", scores[2])

print("RMSE: ", np.mean(rmseList), "MAE: ", np.mean(maeList))

```

```python
#####################################   plot   #####################################
history_dict = history.history
history_dict.keys()

val_loss = history_dict['val_loss']
loss = history_dict['loss']
rmse = history_dict['rmse']
mae = history_dict['mae']
epochs = range(1, len(loss) + 1)

plt.plot(epochs, loss, label='loss')
plt.plot(epochs, val_loss, label='val_loss')
plt.plot(epochs, rmse, label='rmse')
plt.plot(epochs, mae, label='mae')
plt.title('Metrics Graph')
plt.xlabel('Epochs')
plt.ylabel('Value')
plt.legend()
plt.show()
```

Κώδικας εκπαίδευση δικτύου με 2 κρυφά επίπεδα

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold
import keras
from keras.layers import Dense
from keras import backend as K
from keras.callbacks import EarlyStopping
from keras.regularizers import l1
import matplotlib.pyplot as plt

####################################   parameters   ####################################
kfold_value = 5  # number of k for kfold CV
epochs_value = 50  # number of epochs
internal_neurons_number_1 = 14  # number of neurons and internal nodes for first level
internal_neurons_number_2 = 13  # number of neurons and internal nodes for second level
min_MAE_distance = 0.0005  # early stopping when MAE(n)-MAE(n-1) < min_MAE_distance
learning_rate_value = 0.05  # value for lr parameter
momentum_value = 0.6  # value for momentum parameter
regularizer_value = 0  # value for regularizer

####################################   Read dataset   ####################################
dataset = pd.read_csv('dataset/ml-100k/u.data', header=None, delimiter="\t", names=['userId', 'itemId', 'rating', 'timestamp'])
dataset = dataset.apply(pd.to_numeric)

####################################   data_centering   ####################################
average_user_rating = {}
for userId in dataset.userId.unique():
    average_user_rating[userId] = dataset.loc[dataset['userId'] == userId, 'rating'].mean()  # centering
    dataset.loc[dataset['userId'] == userId, 'rating'] -= average_user_rating[userId]
# print(max(dataset.loc[:, 'rating']))
# print(min(dataset.loc[:, 'rating']))
####################################   missing values   ####################################
# NxM_dataset: NxM Array with user ratings and 0 for missing values
NxM_dataset = pd.pivot_table(dataset, index='userId', columns='itemId', values='rating', fill_value=0)
NxM_dataset = NxM_dataset.apply(pd.to_numeric)

####################################   data_normalization   ####################################
for userId in dataset.userId.unique():
    NxM_dataset.loc[userId, :] = NxM_dataset.loc[userId, :] / max(abs(NxM_dataset.loc[userId, :]))

#################################   5-fold_cross_validation   #################################
X = np.array(pd.get_dummies(dataset.userId.unique()))  # input vector
Y = np.array(NxM_dataset.loc[:, :])  #outputvector

kf = KFold(n_splits=kfold_value, shuffle=True)

####################################   A-NN_model   ####################################
rmseList = []  # list for RSME metric
maeList = []  # list for MAE metric

fold = 0
for (train_index, test_index) in kf.split(X):
    fold += 1
    model = keras.Sequential()

    model.add(Dense(len(X), activation="sigmoid",  activity_regularizer=l1(regularizer_value), input_dim=len(X)))  # input level
    model.add(Dense(internal_neurons_number_1, activation="relu", activity_regularizer=l1(regularizer_value), input_dim=len(X)))  # internal level 1
    model.add(Dense(internal_neurons_number_2, activation="relu", activity_regularizer=l1(regularizer_value), input_dim=internal_neurons_number_1))  # internal level 2
    model.add(Dense(len(Y[0]), activation="sigmoid",  input_dim=internal_neurons_number_2))  # output level

    # rmse calculation function
    def rmse(y_true, y_pred):
        return K.sqrt(K.mean(K.square(y_pred - y_true)))

    # mae calculation function
    def mae(y_true, y_pred):
        return K.mean(K.abs(y_pred - y_true))

    keras.optimizers.SGD(lr=learning_rate_value, momentum=momentum_value, decay=0.0, nesterov=False)
    model.compile(loss='mean_squared_error', optimizer='nadam', metrics=[rmse, mae])

    early_stopping = EarlyStopping(monitor='mae', mode='min', min_delta=min_MAE_distance, verbose=1)
    history = model.fit(X[train_index], Y[train_index], validation_data=(X[test_index], Y[test_index]), epochs=epochs_value, batch_size=len(X), callbacks=[early_stopping], verbose=0)

    scores = model.evaluate(X[test_index], Y[test_index], verbose=0)
    rmseList.append(scores[1])
    maeList.append(scores[2])

    print("Fold :", fold, " RMSE:", scores[1], " MAE:", scores[2])

print("RMSE: ", np.mean(rmseList), "MAE: ", np.mean(maeList))

####################################   plot   ####################################
history_dict = history.history
history_dict.keys()

val_loss = history_dict['val_loss']
loss = history_dict['loss']
rmse = history_dict['rmse']
mae = history_dict['mae']
epochs = range(1, len(loss) + 1)

plt.plot(epochs, loss, label='loss')
plt.plot(epochs, val_loss, label='val_loss')
plt.plot(epochs, rmse, label='rmse')
plt.plot(epochs, mae, label='mae')
plt.title('Metrics Graph')
plt.xlabel('Epochs')
plt.ylabel('Value')
plt.legend()
plt.show()
```

Πηγές - Αναφορές

1. https://grouplens.org/datasets/movielens/100k/

2. https://www.jetbrains.com/pycharm/

3. https://www.python.org/downloads/release/python-382/

4. Hyndman, Rob J.; Koehler, Anne B. (2006). "Another look at measures of forecast accuracy". International Journal of Forecasting. 22 (4): 679–688. doi:10.1016/j.ijforecast.2006.03.001

5. Pontius, Robert; Thontteh, Olufunmilayo; Chen, Hao (2008). "Components of information for multiple resolution comparison between maps that share a real variable". Environmental Ecological Statistics. 15: 111–142

6. K. Potdar, T. S., and C. D., “A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers,” IJCA, vol. 175, no. 4, pp. 7–9, Oct. 2017, doi: 10.5120/ijca2017915495.

7. A. F. Agarap, “Deep Learning using Rectified Linear Units (ReLU),” arXiv:1803.08375 [cs, stat], Feb. 2019.

8. J. F. Kros, M. Lin, and M. L. Brown, “Effects of the neural network s-Sigmoid function on KDD in the presence of imprecise data,” Computers & Operations Research, vol. 33, no. 11, pp. 3136–3149, Nov. 2006,

9. https://keras.io/callbacks/

10. T. Dozat, “Incorporating Nesterov Momentum into Adam,” p. 6.

11. Σ. Λυκοθανάσης, Ε. Γεωργόπουλος, “ΥΠΟΛΟΓΙΣΤΙΚΗ ΝΟΗΜΟΣΥΝΗ, ΠΑΝΕΠΙΣΤΗΜΙΑΚΕΣ ΣΗΜΕΙΩΣΕΙΣ”, Πανεπιστήμιο Πατρών, Πάτρα 2014

12. A. Y. Ng, “Feature selection, L 1 vs. L 2 regularization, and rotational invariance,” in Twenty-first international conference on Machine learning - ICML ’04, Banff, Alberta, Canada, 2004, p. 78