Εργασία 4, Επίλυση προβλήματος παλινδρόμησης με χρήση Radial-Basis Function δικτύου

Νίκος Λαδιάς

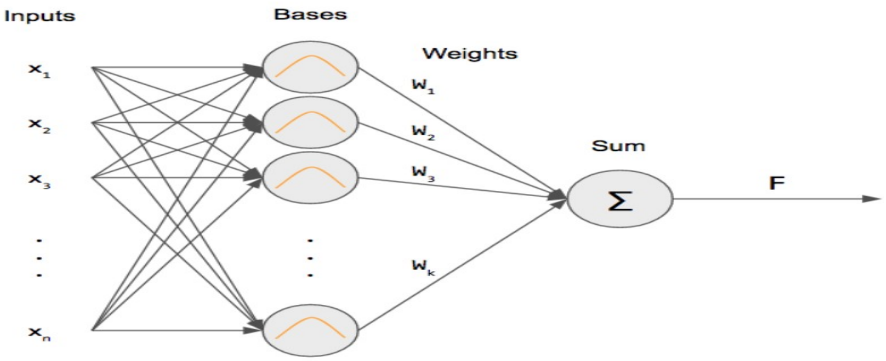
Αύγουστος 2021

Η εργασία είναι υλοποιημένη σε ένα αρχείο .ipynb με την βοήθεια του Jupyter Notebook , όπου μας δίνεται η δυνατότητα να εκτελούμε κελιά κώδικα κάθε φορά και αν βλέπουμε τα αντίστοιχα μυνήματα της κονσόλας ακριβώς κάτω απο το κάθε κελί. Αυτό μας προσφέρει καλύτερη οπτικοποίηση των αποτελεσμάτων και οργάνωση του κώδικα. Σε κάθε κελί υπάρχουν και τα αντίστοιχα σχόλια (στα Αγγλικά) επεξήγησης του κώδικα.

Περιεχόμενα

1 Εισαγωγή στα δίκτυα RBF

Ένα δίκτυο RBF είναι παρόμοιο με ένα δίκτυο 2 στρωμάτων. Έχουμε μια είσοδο που συνδέεται πλήρως με ένα κρυφό επίπεδο. Στη συνέχεια, παίρνουμε την έξοδο του κρυφού στρώματος και εκτελούμε ένα σταθμισμένο άθροισμα για να πάρουμε την έξοδό μας.



Εικόνα 1: RBF δίκτυο

Ένα RBF δίκτυο στο κρυφό του στρώμα μετασχηματίζει την είσοδο με βάση μια Radial Basis Function. Για την ακρίβεια, χρησιμοποιούμε μια Gaussian RBF. Ενδιαφερόμαστε κυρίως για τις ιδιότητες της καμπύλης "καμπάνας" της Γκαουσιανής. Μπορούμε να χρησιμοποιήσουμε έναν γραμμικό συνδυασμό Γκαουσιανών για να προσεγγίσουμε οποιαδήποτε συνάρτηση. Όταν παίρνουμε το άθροισμα, παίρνουμε μια συνεχή συνάρτηση. Για να το κάνουμε αυτό, πρέπει να γνωρίζουμε πού να τοποθετήσουμε τα γκαουσιανά κέντρα c_j και τις τυπικές αποκλίσεις τους σ_j.

Μπορούμε να χρησιμοποιήσουμε την ομαδοποίηση k-means στα δεδομένα εισόδου μας για να βρούμε πού να τοποθετήσουμε τις Γκαουσιανές. Το σκεπτικό πίσω από αυτό είναι ότι θέλουμε **οι Γκαουσιανές μας να "καλύπτουν" τις μεγαλύτερες συστάδες δεδομένων, αφού έχουν αυτό το χαρακτηριστικό σχήμα καμπάνας.** Σε αυτή την εργασία τα γκαουσιανά κέντρα c_j υπολογίζονται με χρήση του K-means αλγορίθμου ομαδοποίησης ενώ η διασπορά κάθε νευρώνα παίρνει τιμή $\sigma_j = \sigma \forall j$, όπου $\sigma = \frac{d_{max}}{\sqrt{2k}}$ όπου d_{max} είναι η μέγιστη απόσταση μεταξύ δύο κέντρων συστάδων(clusters) και k είναι ο αριθμός των κέντρων των συστάδων.

2 Το dataset

Το Boston housing dataset, ενσωματώνεται για την εκπόνηση της εργασίας μέσω της βιβλιοθήκης skicit learn. Αφού ενσωματωθούν γενικά όλες οι απαραίτητες βιβλιοθήκες και κλάσεις , δημιουργείται ένα panda dataframe για το Boston housing dataset. Με αυτή τη μορφή βλέπουμε το dataset σε δισδιάστατη δομή δεδομένων σε μορφή πίνακα με δυνατότητα μεταβολής μεγέθους,με επισημασμένους άξονες (γραμμές και στήλες).

Έπειτα, αφού προστεθεί στις μεταβλητές του dataframe η 'PRICE' μέσω της target variable του boston_dataset. Αυτή η μεταβλητή μας ενδιαφέρει άμεσα καθώς αναγράφονται σε αυτήν οι τιμές των ακινήτων που θέλουμε να προβλέψουμε.

Με την describe() χρησιμοποιείται για την προβολή ορισμένων βασικών στατιστικών στοιχείων, όπως εκατοστημόριο, μέσος όρος, τυπική απόκλιση κ.λπ. για δεδομένο dataframe όπως και του Boston housing dataset. Ελέγχουμε αν υπάρχουν ελλείπουσες τιμές στα δεδομένα. Μετράμε τον αριθμό των ελλιπών τιμών για κάθε χαρακτηριστικό χρησιμοποιώντας την .isnull(). Το σύνολο δεδομένων χωρίζεται , όπως ζητείται, σε υποσύνολα εκπαίδευσης και δοκιμής, με τη δεδομένη αναλογία 75-25%. Επίσης γίνεται και η παρακράτηση του 20% των δεδομένων εκπαίδευσης που ζητείται για επικύρωσης.

3 Υλοποίηση δικτύου RBF

Αρχικά , ορίζεται η κλάση αρχικοποίησης InitCentersKMeans, η οποία κληρονομεί απο την οικογένεια των Initializers του keras, ενώ ορίζεται και μια custom-function kmeans, η οποία θα χρησιμοποιηθεί για την αρχικοποίηση των διασπορών των νευρώνων του δικτύου. Η κλάση αυτή περιέχει τις συναρτήσεις init(), δηλαδή τον constructor της που αρχικοποιεί τις μεταβλητές της κλάσης, καθώς και την call(). Η call(), χρησιμοποιεί την μέθοδο KMeans() της βιβλιοθήκης sklearn ,

Η κύρια δομή όπως είδαμε , ενός RBF δικτύου, είναι το κρυφό στρώμα μετασχηματισμού της εισόδου. Για να υλοποιηθεί το δίκτυο με αυτό το τρόπο, μια κλάση RBFLayer δημιουργείται που κληρονομεί απο την οικογένεια Layer απο την tensorflow.keras.layers βιβλιοθήκη. Η κλάση αυτή μοντελοποιεί το κρυφό (και πρώτο) στρώμα του δικτύου χρησιμοποιώντας τις μεθόδους :

1. init(): Constructor της κλάσης, αρχικοποιεί τις μεταβλητές ενώ επίσης καλέι και τον constructor της "γονικής" κλάσης με την μέθοδο super.

2. build(): Δείνει τιμές στα κέντρα και στα βάρη του RBF δικτύου. Καλεί και την build() της γονικής με super().

3. initialize_stds(): Χρησιμοποιείται για τον υπολογισμό της d_{max} μέγιστης απόστασης μεταξύ δύο κέντρων (εδώ χρησιμοποείται η custom function kmeans που ορίστηκε νωρίτερα για τον υπολογισμό των κέντρων). Έτσι θέτει την διασπορά κάθε νευρώνα ίση με $\frac{d_{max}}{\sqrt{2k}}$

4. call(): Η λογική για την εκτέλεση του επιπέδου θα πρέπει να είναι call(x). Αυτό δέχεται τον τανυστή εισόδου και επιστρέφει τον τανυστή εξόδου. Τα αντικείμενα τανυστών είναι ο τρόπος με τον οποίο το keras χειρίζεται την είσοδο και την έξοδο των επιπέδων. Έχουν έναν ορισμένο αριθμό καθορισμένων διαστάσεων (μέγεθος εισόδου) και έναν ορισμένο αριθμό μη καθορισμένων διαστάσεων (το batch size και οποιαδήποτε άλλη ευέλικτη διάσταση). Οι Tensors (και τα αντικείμενα Variable) είναι ένα είδος ασύγχρονης υπόσχεσης που δεν υλοποιείται εν μέρει μέχρι τη στιγμή της μεταγλώττισης και δεν υλοποιείται πλήρως μέχρι τη στιγμή της εκτέλεσης. Οι αριθμητικές πράξεις με τανυστές γίνονται με την βοήθεια του keras backend (from keras import backend as K).

5. compute_output_shape(): Αυτή η μέθοδος θα πρέπει να καθορίζει το σχήμα εξόδου για ένα δεδομένο σχήμα εισόδου. Χρησιμοποιείται κατά τη διάρκεια του χρόνου μεταγλώττισης για να επικυρώσει ότι τα σχήματα των τανυστών που περνούν μεταξύ των επιπέδων είναι σωστά.

6. get_config: Πρέπει να οριστεί έτσι ώστε να μπορούμε να χρησιμοποιήσουμε το model_from_json.

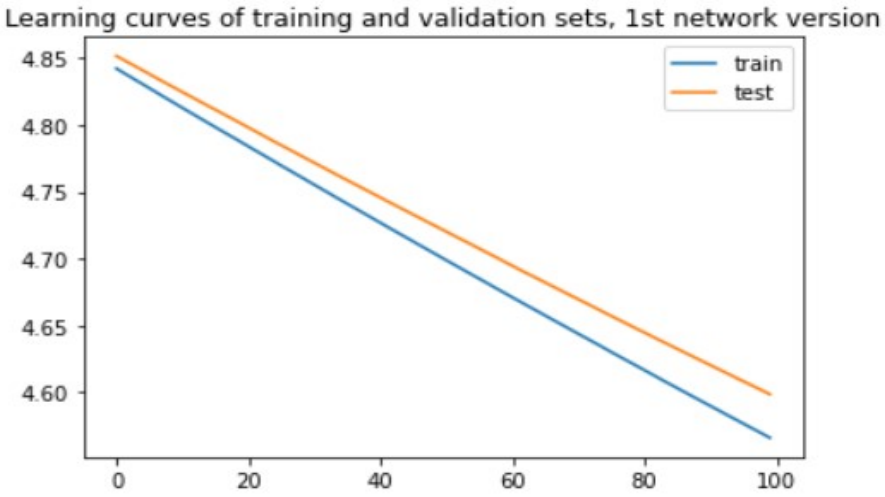
Αφού έχουν λοιπόν οριστεί οι απαραίτητες κλάσεις και συναρτήσεις για να μπορούμε να μοντελοποιήσουμε το RBF στρώμα, η δημιουργία ενός RBF δικτύου είναι πλέον εύκολη. Με τον ίδιο τρόπο που χρησιμοποιούμε το keras της tensorflow, χτίζουμε το δίκτυο κανονικά ως γνωστόν. Αφού γνωρίζουμε τους νευρώνες του κρυφού RBF στρώματος, του στρώματος εξόδου, καθώς και τον τύπο του optimizer . Ορίζουμε αρχικά το RBF στρώμα, αρχικοποιούμε τις διασπορές των νευρώνων, ορίζουμε το μοντέλο του δικτύου ως tf.keras.Sequential(). Προσθέτουμε το στρώμα RBF και έπειτα το στρώμα εξόδου και το δίκτυο είναι έτοιμο . Έτσι μπορούμε εύκολα να κάνουμε compile το μοντέλο , θα χρησιμοποιήσουμε για μετρική κόστους το "mean squared error", δηλαδή το μέσο τετραγωνικό σφάλμα. Τέλος κάνουμε fit το μοντέλο στα δεδομένα εκπαίδευσης, περνώντας και τα δεδομένα επικύρωσης και ορίζοντας την εκπαίδευση για 100 εποχές. Η συνάρτηση ορίζεται η SparseCategoricalCrossEntropy. Δημιουργούνται και εκπαιδεύονται 3 διαφορετικά μοντέλα για τις τρείς περιπτώσεις δικτύων που ζητούνται (διαφορετικός αριθμός νευρώνων ενδιάμεσου RBF κρυφού στρώματος), των οποίων τα αποτελέσματα αναλύονται .

4 Αποτελέσματα και Συμπεράσματα

4.1 Αποτελέσματα

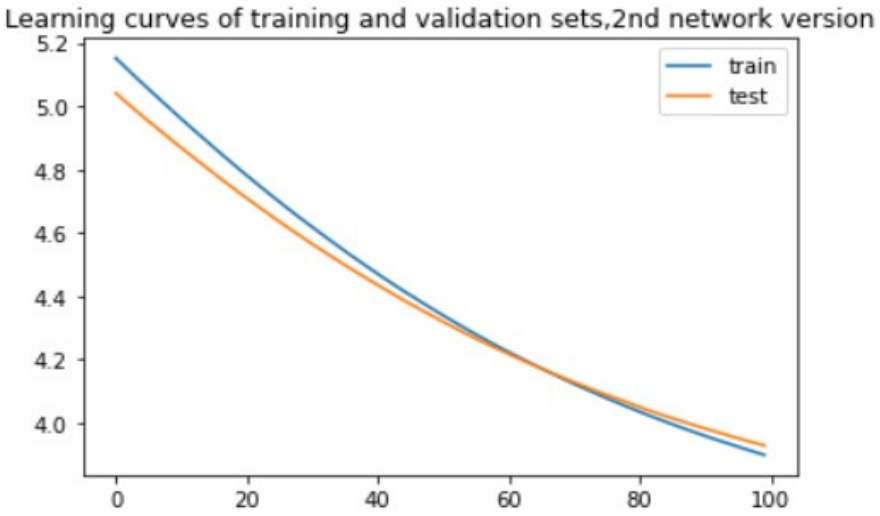
Για κάθε ζητούμενο αριθμό νευρώνων του κρυφού RBF στρώματος , παράγονται καμπύλες εκμάθησης για training και validation δεδομένα, καθώς και οι μετρικές αξιολόγησης R^2 και RMSE. Τα παρακάτω αποτελέσματα προέκυψαν μετά απο συνεχόμενες προσπάθειες fitting του αντίστοιχου μοντέλου.

Για αριθμό νευρώνων του κρυφού RBF στρώματος ίσο με το 10% του πλήθους των δεδομένων εκπαίδευσης, παρατηρήθηκε πως η μετρική αξιολόγησης R^2 που προκύπτει έχει μια τιμή αρνητική (πολλές φορές και αρκετά πιο αρνητική σε σύγκριση με τα υπόλοιπα μοντέλα) και ένα RMSE της τάξεως του 20-50. Εδώ να σημειωθεί πως το R^2 είναι αρνητικό, εάν το επιλεγμένο μοντέλο ταιριάζει χειρότερα από μια οριζόντια γραμμή. Όσο αφορά τις καμπύλες εκμάθησης, αυτές μοιάζουν περισσότερο με δύο ευθείες με ένα αρκετά μικρό παρατηρήσιμο κενό μεταξύ τους.



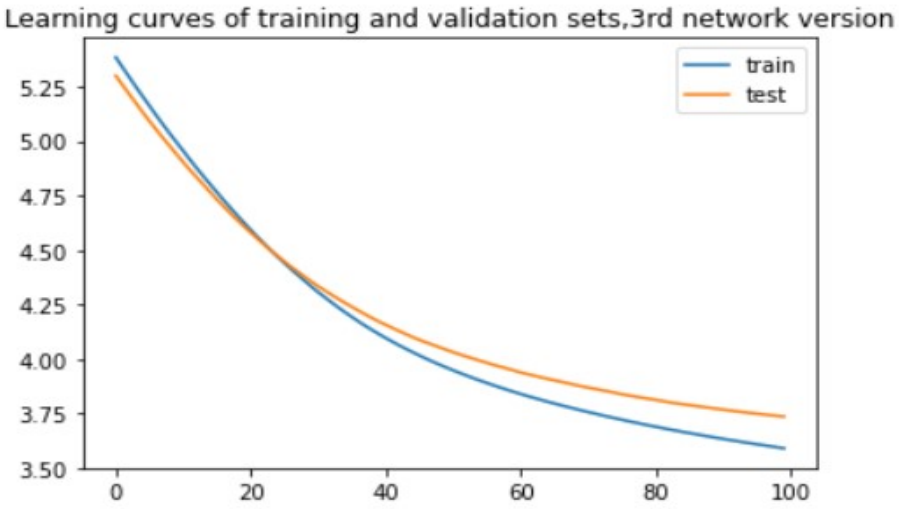
Εικόνα 2: Ενδεικτικύες καμπύλες εκμάθησης για αριθμό νευρώνων RBF στρώματος 10% του πλήθους των δεδομένων εκπαίδευσης

Για αριθμό νευρώνων του κρυφού RBF στρώματος ίσο με 50% των δεδομένων εκπαίδευσης , για τις μετρικές αξιολόγησης παρατηρήθηκαν τα εξής. Η μετρική αξιολόγησης R^2, παίρνει πλέον θετικές τιμές κοντά στο μηδέν συνήθως ενώ πιο σπάνια μπορεί να πάρει και αρνητική τιμή πολύ κοντά στο μηδέν όμως πάλι. Η μετρική αξιολόγησης RMSE δεν παίρνει πολύ μεγάλες τιμές (ενδεικτικά παρατηρήθηκε να κυμαίνεται στο εύρος 7-20 περίπου). Οι καμπύλες εκμάθησης είναι πλέον καμπύλες και όχι ευθείες απο ένα σημείο της εκπαίδευσης και μετά (εποχή 60 και μετά περίπου) ταυτίζονται ή βρίσκονται αρκετά κοντά.



Εικόνα 3: Ενδεικτικύες καμπύλες εκμάθησης για αριθμό νευρώνων RBF στρώματος 50% του πλήθους των δεδομένων εκπαίδευσης

Τέλος, για αριθμό νευρώνων του κρυφού RBF στρώματος ίσο με 90% των δεδομένων εκπαίδευσης, τα αποτελέσματα έχουν ως εξής. Η μετρικής αξιολόγησης R^2, πλέον παίρνει κατά κανόνα θετική τιμή , αρκετά κοντά στο μηδέν (0.2-0.3 συνήθως) ενώ αρκετά πιο σπάνια θα πάρει αρνητική τιμή και αυτή κοντά πάλι στο μηδέν. Η μετρική αξιολόγησης RMSE, έχει πλέον μειωθεί σημαντικά παίρνωντας τιμές στο εύρος 7-12. Οι καμπύλες εκμάθησης μοιάζουν πολύ με την προηγούμενη περίπτωση , χωρίς μεγάλες διαφορές. Είναι μια παρόμοια καμπύλη η οποία όμως στρέφει τα κοίλα προς τα κάτω περισσότερο κατα την διάρκεια της εκπαίδευσης, αυτό ισχύει και για τις δύο καμπύλες training και validation δεδομένων.



Εικόνα 4: Ενδεικτικύες καμπύλες εκμάθησης για αριθμό νευρώνων RBF στρώματος 90% του πλήθους των δεδομένων εκπαίδευσης

Τα αποτελέσματα και τα διαστήματα που δόθηκαν για τις τιμές των μετρικών αξιολογήσεων δεν προέκυψαν απο κάποια στατιστική ανάλυση. Είναι όμως αρκετά κοντά στα πραγματικά διαστήματα καθώς παράχθηκαν απο παρατήρηση μετά απο αλλεπάλληλες προσπάθειες fitting του αντίστοιχου μοντέλου, ενώ αποδίδουν μια καλή εικόνα για την επίδοση του κάθε μοντέλου.

4.2 Συμπεράσματα

Αρχικά υπενθυμίζεται πως "κακό" RMSE είναι το μεγαλύτερο σε τιμή RMSE, ενώ "κακό" R^2 είναι ένα αρνητικό R^2. Απο τα παραπάνω προκύπτει σχεδόν ξεκάθαρα πως μεγαλύτερος αριθμός νευρώνων του κρυφού στρώματος φαίνεται να προσφέρει καλύτερη επίδοση του μοντέλου. Η μετρική αξιολόγησης με την οποία εκπαδεύτηκαν τα δίκτυα ήταν η ακρίβεια (accuracy). **Άρα μπορούμε να ισχυριστούμε πως έχουμε μεγαλύτερη ακρίβεια στο μοντέλο μας για μεγαλύτερο αριθμό νευρώνων του κρυφού στρώματος.** Τώρα καθώς αυτό δεν φαίνεται να είναι απόλυτο , αφού υπήρχαν περιπτώσεις όπου οι μετρικές R^2, RMSE ήταν "χειρότερες" για μοντέλο με λιγότερους νευρώνες, όμως αυτό προέκυπτε αρκετά πιο σπάνια και δεν φαινεται να είναι τόσο συχνό ώστε να μην μπορεί κανείς να δηλώσει πως μεγαλύτερος αριθμός νευρώνων συνεπάγεται μεγαλύτερη ακρίβεια. Να σημειωθεί πως overfitting στα δεδομένα μας δεν παρατηρήθηκε σε καμία περίπτωση, πράγμα θετικό για την εκπαίδευση του δικτύου με κάποιο απο τα μοντέλα που δημιουργήθηκαν.

Τέλος αξίζει να σημειωθεί πως υπάρχει και ένας ακόμη παράγοντας που παρατηρήθηκε να επηρεάζει την απόδοση του εκάστοτε μοντέλου. Αυτός είναι η αρχική τιμή που

δίνεται για τα betas στην δήλωση του RBF στρώματος. Για μεγαλύτερες τιμές αυτής της αρχικής τιμής (π.χ. 10) η απόδοση χειροτέρευε αρκετά καθώς το μοντέλο χρειάζεται απο ότι φαίνεται παραπάνω εκπαίδευση για μεγαλύτερες αρχικές τιμές των βαρών του για να φτάσει σε παρόμοια απόδοση. **Η τιμή που δίνεται για τα αρχικά βάρη του RBF στρώματος είναι η ίδια σε όλα για λόγους ισοδύναμιας των δικτύων**, η τιμή επιλέχτηκε να είναι 2.0 .

5 Fine tuning RBF δικτύου

Στο συγκεκριμένο κομμάτι της εργασίας, σκοπός είναι η εκτέλεση εύρεση των βέλτιστων τιμών για μερικές υπερπαραμέτρους του δικτύου, και η τελική εκπαίδευση και αξολόγηση ενός μοντέλου με βάση τις επιλεγμένες τιμές. Το RBF δίκτυο που θα εξεταστεί στο παρόν κομμάτι θα έχει την ίδια αρχιτεκτονική με τα παραπάνω μοντέλα (ένα RBF κρυφό στρώμα και ένα γραμμικό στρώμα εξόδου), με την επιπλέον προσθήκη drop out κανονικοποίησης για τους νευρώνες του στρώματος εξόδου.

Ου υπερπαράμετροι προς εξέταση είναι :

1. αριθμός νευρώνων RBF κρυφού στρώματος : $n_{h1} \in \{5\%, 15\%, 30\%, 50\%\}$
2. αριθμός νευρώνων κρυφού στρώματος εξόδου : $n_{h2} \in \{32, 64, 128, 256\}$
3. dropout πιθανότητα : $p \in \{0.2, 0.35, 0.5\}$

Η αρχικοποίηση των διασπορών των νευρώνων του RBF στρώματος γίνεται όπως και στο πρώτο τμήμα. Η εκπαίδευση του συνολικού δικτύου γίνεται σε δύο φάσεις, όπως και πρίν, με χρήση του Kmeans για το πρώτο στρώμα και SGD optimizer ($\text{lr} = 0.001$, 100 εποχές) για το δεύτερο στρώμα. Ως συνάρτηση κόστους θεωρείται το μέσο τετραγωνικό σφάλμα και ως μετρική αξιολόγησης το RMSE.

Η διαδικασία εύρεσης των βέλτιστων τιμών των παραμέτρων γίνεται με τη χρήση ενός διαθέσιμου framework (το keras-tuner), ενώ το 20% των δεδομένων εκπαίδευσης παρακρατείται για επικύρωση. Αφού οριστεί η συνάρτηση που βοηθάει στο "χτίσιμο" του επιθυμητού μοντέλου με βάση : τις δεδομένες ιδιότητες του δικτύου , την βοήθεια του keras, το RBF στρώμα μας (που ορίζεται ακριβώς όπως και στο προηγούμενο κομμάτι της εργασίας). Το μοντέλο χτίζεται όπως και πρίν με μόνη διαφορά την επιπλέον προσθήκη του dropout στρώματος που επικάθεται στο στρώμα εξόδου του δικτύου. Η αρχικοποίηση των διασπορών των νευρώνων όπως και πρίν με την custom συνάρτηση initialize_stds().

Η διαδικασία tuning δεν έχει κάποια διαφορά με το προηγούμενο σκέλος της εργασίας. Χρησιμοποιείται πάλι κανονικά keras tuner, με χρήση της Hyperband κλάσης για να γίνει το fit του μοντέλου. Δεν έχει χρησιμοποιηθεί κάποιο early stopping καθώς δεν είναι μεγάλος και ο ζητούμενος αριθμός εποχών εκπαίδευσης. Μετά το πέρας της διαδικασίας του search για τις καλύτερες υπερπαραμέτρους

του δικτύου γίνεται και ένα fit του παραγόμενου με αυτές μοντέλου για 200 εποχές, έτσι ώστε να υπάρχει και μια οπτικοποίηση μετά των αποτελεσμάτων του μοντέλου αυτού (καμπύλες εκμάθησης για training και validation set), το μοντέλο φάνηκε να αποκρίνεται καλά χωρίς να παρατηρείται κάποιο overfitting . Να σημειωθεί πως στις μετρικές αξιολόγησεις κατα το compile ορίστηκε και η ζητούμενη root_mean_squared_error, αλλά και η accuracy, διότι δεν ήταν δυνατή η αναγνώριση του objective στην δήλωση του Hyperband tuner. Εφόσον το objective ήταν το val_accuracy, χρειαζόταν και την μετρική του accuracy για να προχωρήσει σωστά η διαδικασία.