ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

ΒΑΘΙΑ ΜΑΘΗΣΗ

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: Πορλού Χάιδω

ΣΧΟΛΗ: ΗΜΜΥ ΑΠΘ

ΑΕΜ: 9372

ΕΡΓΑΣΙΑ 3

RADIAL BASIS FUNCTION (RBF)

Εισαγωγή

Η 3η (και τελευταία) εργασία που καλούμαστε να ολοκληρώσουμε στο μάθημα Νευρωνικά Δίκτυα – Βαθιά Μάθηση αφορά (μετά από δική μας επιλογή ανάμεσα σε 3 υλοποιήσεις) ένα πρόβλημα προσέγγισης με τη χρήση ενός Radial Basis Function (RBF) neural network.

Σαν dataset επιλέχθηκε το California Housing Dataset, το οποίο αποτελείται από 20,640 δείγματα και 9 features (8 κλασσικά και 1 target feature). Το dataset αυτό περιλαμβάνει διάφορες πληροφορίες (ή μέση τιμή πληροφοριών) για σπίτια στην περιοχή της Καλιφόρνια (π.χ. ηλικία σπιτιού και μέσο αριθμό υπνοδωματίων), μέσω των οποίων προβλέπεται η τελική μέση τιμή ενός σπιτιού (το target feature παίρνει τιμές στο διάστημα [0.15 – 5] οι οποίες πολλαπλασιάζονται με το 100,000 για να δοθεί η τελική τιμή). Ο χωρισμός των δεδομένων, όπως αναφέρεται και στην εκφώνηση, έγινε τυχαία με σύνολο εκπαίδευσης ίσο με το 60% και σύνολο ελέγχου ίσο με το 40%.

Οι τιμές των features είναι όλες numerical, και έτσι προχωράμε στη χρήση τους σε ένα πρόβλημα προσέγγισης (regression). Λόγω της κακής εκπαίδευσης και πρόβλεψης διαφορετικά, προχωράμε στην κανονικοποίηση των δεδομένων με το z-score, το οποίο παρουσιάζεται ως εξής: $z = (X – \mu) / \sigma$. Στον τύπο αυτό X είναι μια αρχική τιμή, μ είναι η μέση τιμή όλων των δεδομένων και σ είναι υ τυπική απόκλισή τους. Με την κανονικοποίηση έχουμε πλέον μικρότερες τιμές πιο κοντά στο 0, οπότε λειτουργεί καλύτερα και οποιοδήποτε μοντέλο regression χρησιμοποιήσουμε.

Όπως αναφέρθηκε και πιο πάνω, το μοντέλο που επιλέχθηκε να χρησιμοποιηθεί είναι ένα Radial Basis Function neural network. Ακολουθεί μια σύντομη περιγραφή της λειτουργίας του. Αρχικά, σημειώνεται πως το μοντέλο που χρησιμοποιήθηκε αποτελείται από το απλούστερο δυνατό RBF δίκτυο και δεν περιλαμβάνει κάποιο MLP όπως συνηθίζεται για μια πιο ολοκληρωμένη μελέτη. Η επιλογή αυτή έγινε αφενός για να μελετηθεί εις βάθος το RBF, και αφετέρου για να παρουσιαστούν αποτελέσματα που αντιστοιχούν στην πραγματικότητα και δεν εμφανίζουν πολύ καλές μετρικές λόγω των άλλων κομματιών ενός regression μοντέλου (π.χ. άλλα layers).

Ένα radial basis function δίκτυο αποτελείται, στο εσωτερικό του στρώμα, από μια συνάρτηση ενεργοποίησης ακτινικού τύπου. Πιο περιγραφικά, ένα δίκτυο RBF αποτελείται από μια είσοδο (μεγέθους ίσου με τα δεδομένα εισόδου), ένα εσωτερικό στρώμα RBF, και ένα στρώμα εξόδου που στην περίπτωση μας έχει μόνο ένα νευρώνα εξόδου (πρόβλημα regression). Η πληροφορία περνάει δηλαδή από την radial basis συνάρτηση ενεργοποίησης και προχωρά στην έξοδο, παρόμοια με τη διαδικασία που χρησιμοποιείται σε ένα MLP.

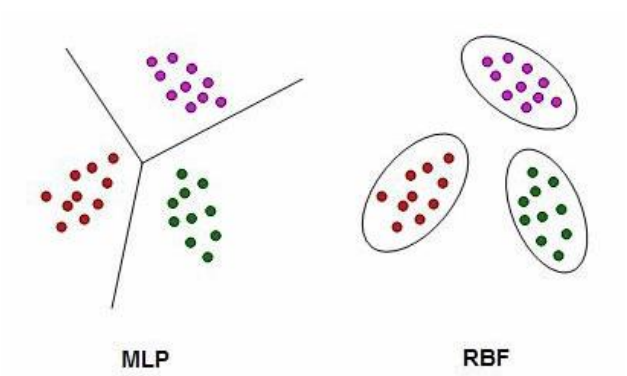
Η Radial Basis Function ως layer

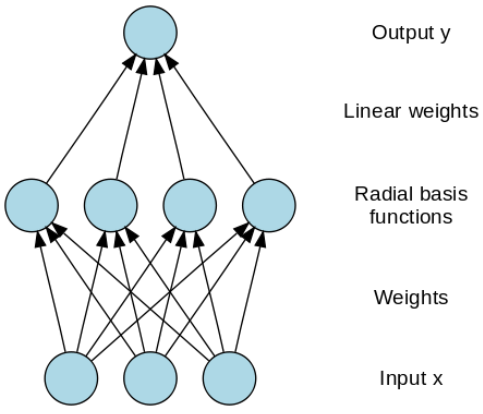
Ο τρόπος με τον οποίο εισάχθηκε η Radial Basis Function στο μοντέλο μας, ώστε να δημιουργηθεί το αντίστοιχο network, είναι με τη δημιουργία ενός custom layer. Το layer

αυτό χρησιμοποιείται με τον ίδιο τρόπο που χρησιμοποιούνται τα «έτοιμα» layers στο keras (π.χ. dense layer), με τη διαφορά πως εμείς ορίζουμε τις μεταβλητές εισόδου και τη συμπεριφορά του με τη συγγραφή κώδικα. Παρακάτω αναλύεται το custom layer που δημιουργήσαμε.

Η είσοδος του RBF layer αποτελείται από: τον αριθμό των νευρώνων εξόδου, το gamma, και τον initializer των βαρών. Ο αριθμός των νευρώνων εξόδου είναι, αυτονόητα, ένας αριθμός ο οποίος καθορίζει το μέγεθος του layer και πόσες τιμές θα μας επιστρέψει. Το gamma, το οποίο θα αναλυθεί και στη συνέχεια, επηρεάζει το πόσο αυστηρά κάνει το μοντέλο μας fit στα δεδομένα (όπως αναφέρεται και παρακάτω, μεγάλο gamma σημαίνει μικρότερη ακτίνα και μεγαλύτερη αυστηρότητα). Ο initializer, τέλος, προσδιορίζει τον τρόπο με τον οποίο αρχικοποιούνται τα βάρη του layer, ώστε να προχωρήσουμε μετά στην εκπαίδευση τους.

Στο εσωτερικό του κώδικα του layer (συνάρτηση call), περιγράφεται η λειτουργία του δικτύου μας. Η radial basis function που χρησιμοποιήθηκε είναι της μορφής: $\mathbf{f(x)=e^{(-\gamma\sum(x-\mu)^2)}}$. Σχολιάζεται εδώ πως συχνά η συνάρτηση ενεργοποίησης αυτού του τύπου (η οποία πραγματεύεται στην ουσία ακτίνες στο εσωτερικό των οποίων ανήκουν ή όχι τα δεδομένα) στη θέση του γ παρουσιάζει μια τιμή $1/r^2$, όπου r η ακτίνα του αντίστοιχου σχήματος. Το gamma (γ) λοιπόν εδώ είναι αντιστρόφως ανάλογο του τετραγώνου της ακτίνας. Το x στον τύπο αποτελεί την πληροφορία εισόδου, και το μ το σύνολο των βαρών του στρώματος. Βρίσκουμε δηλαδή το άθροισμα των τετραγώνων της απόστασης της εισόδου από τα βάρη, και το πολλαπλασιάζουμε με τη σταθερά γ. Τέλος, υψώνουμε τα αποτελέσματα ως προς e για να πάρουμε την τελική λίστα των τιμών των N νευρώνων εξόδου του RBF layer. Στις παρακάτω εικόνες φαίνεται πως γενικευμένα λειτουργεί μια RBF σε σχέση με ένα MLP.





Σημειώνεται εδώ πως η ακτίνα των διαφόρων σχημάτων που δημιουργούνται σε αυτή τη διαδικασία μπορεί να είναι μεταβαλλόμενη, να επιλέγεται δηλαδή ώστε να ταιριάζει καλύτερα στα δεδομένα που θέλει να περικλείσει (διαφορετικές ακτίνες για διαφορετικούς «κύκλους»). Δεν προχώρησα όμως σε μια τέτοια διαδικασία χάριν απλότητας και λόγω δυσκολίας όσον αφορά το κομμάτι της python.

Μετά την ανάλυση του εσωτερικού του νέου RBF layer, προχωράμε στη μελέτη του μοντέλου που το περικλείει. Στη συνέχεια παρουσιάζονται όσα πειράματα έγιναν με μεταβολή των παρακάτω χαρακτηριστικών/παραμέτρων:

- number of neurons
- gamma
- weight initializer
- optimizer

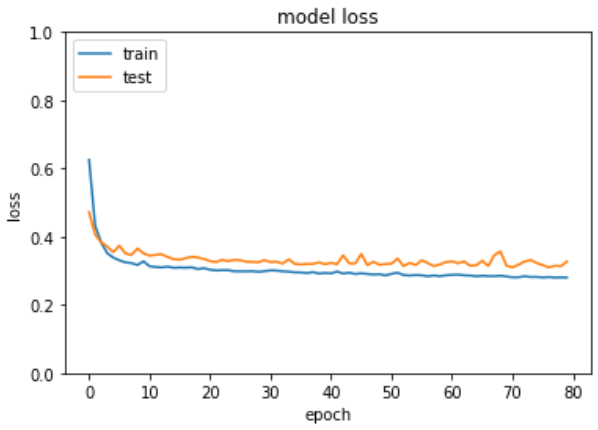
Τα αποτελέσματα σε κάποιες περιπτώσεις ήταν πολύ ενθαρρυντικά, ενώ σε άλλες όχι τόσο. Οι μετρικές που επιλέχθηκαν για την αξιολόγηση του μοντέλου είναι το RMSE (rooted mean square error) και το R^2. Το RMSE δείχνει ένα καλύτερο μοντέλο όσο πιο μικρό είναι (αυτονόητο μιας και παρουσιάζει error) και το R^2 όσο πιο μεγάλο είναι (με μάξιμουμ τιμή το 1 – παρουσιάζει ποσοστό της εισόδου που μπορεί να «εξηγηθεί» σωστά από το μοντέλο).

Σαν πρώτο πείραμα προχωρήσαμε σε αλλαγή του αριθμού των νευρώνων εξόδου του RBF layer. Οι σταθερές τιμές που επιλέξαμε για τις υπόλοιπες παραμέτρους είναι: gamma = 0.1, initializer = KMeans (θα αναλυθεί αργότερα) & optimizer = Adam (με lr=0.001). Σημειώνεται επίσης πως το πλήθος των δεδομένων εισόδου εκπαίδευσης (με 8 features το καθένα) είναι 12,384, οπότε ο αριθμός των νευρώνων θα είναι φυσικά μικρότερος από αυτό.

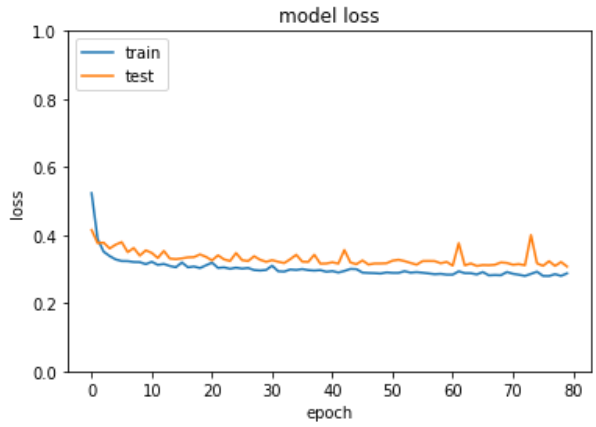
Neurons	Gamma	Initializer	Optimizer	RMSE	R^2	Time (s)
1000	0.1	KMeans	Adam	0.571850161	0.672987392	76.7993
2000	0.1	KMeans	Adam	0.554600216	0.692418599	132.9279
4000	0.1	KMeans	Adam	0.552371055	0.694886217	270.3634
8000	0.1	KMeans	Adam	0.588420736	0.653761037	519.2367
10000	0.1	KMeans	Adam	0.563891950	0.682025868	682.6041
10000	0.1	KMeans	Adam	0.805599752	0.351009038	702.1220

Παρουσιάζονται επίσης τα διαγράμματα του loss function που επιλέχθηκε, δηλαδή του mean squared error, για τα δεδομένα εκπαίδευσης και ελέγχου, ώστε να μελετηθεί και η πιθανότητα υπερεκπαίδευσης.

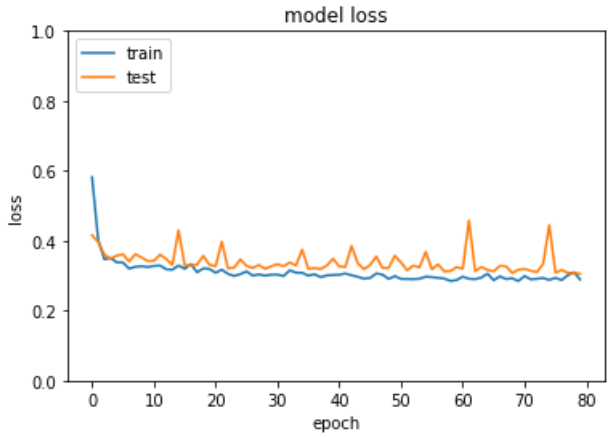
a) 1000 neurons



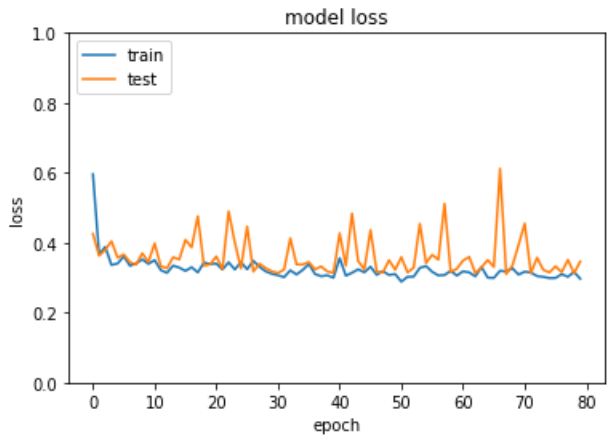
b) 2000 neurons



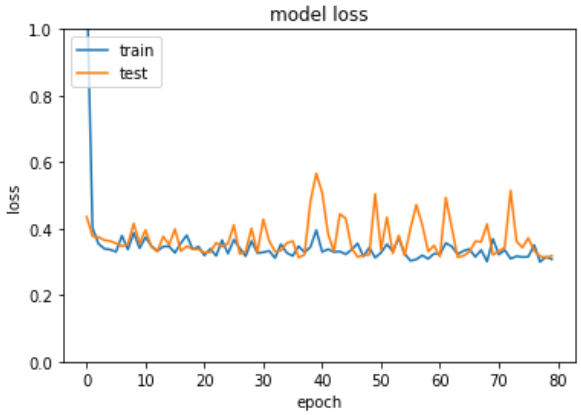
c) 4000 neurons



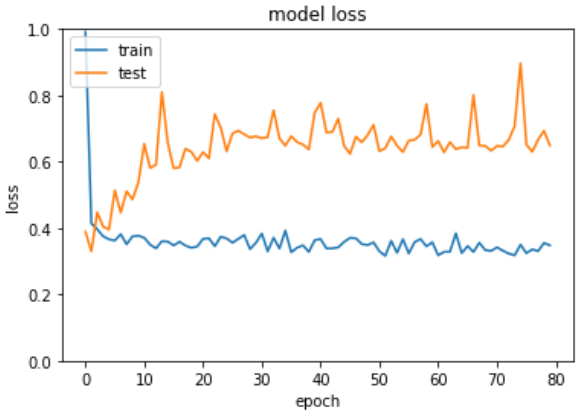
d) 8000 neurons



e) 10000 neurons



f) 10000 neurons (overtraining)



Από τα δεδομένα του πίνακα βλέπουμε πως τα καλύτερα αποτελέσματα (με μικρότερο σφάλμα και μεγαλύτερο R^2) εμφανίζονται όταν έχουμε αριθμό νευρώνων ίσο με 2000 και 4000. Ο χρόνος μεταξύ αυτών βέβαια είναι διαφορετικός, με την περίπτωση των 2000 νευρώνων να είναι αρκετά μικρότερος (132s vs 270s). Θα ήταν δηλαδή καλή επιλογή η έξοδος 2000 νευρώνων, αν και στη συνέχεια κράτησα τους 4000 νευρώνες σαν την default επιλογή.

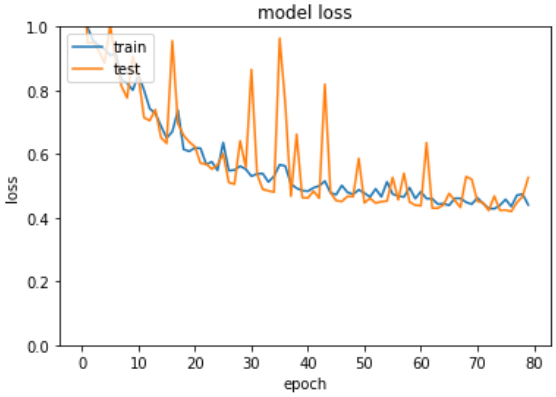
Παρατηρείται πως όσο αυξάνεται ο αριθμός των νευρώνων τόσο αυξάνεται και ο χρόνος εκπαίδευσης, καθώς και οι έντονες διακυμάνσεις στο MSE κυρίως του συνόλου ελέγχου αλλά και εκπαίδευσης. Τέλος, συμπεριέλαβα δύο διαφορετικά πειράματα για την περίπτωση των 10,000 νευρώνων, καθώς περίπου τις μισές φορές που δοκιμάσθηκε το μοντέλο με τόσους πολλούς νευρώνες, κατέληξε σε υπερεκπαίδευση. Ένα παράδειγμα αυτού είναι η τελευταία περίπτωση, όπου τα RMSE και R^2 έχουν πολύ κακές τιμές, και στο σχήμα φαίνεται πως το σφάλμα στο σύνολο ελέγχου αυξάνεται έντονα μετά από ένα σημείο, αποδεικνύοντας την αρχική μας υπόθεση.

Σημειώνεται εδώ πως είναι λογικό με μεγάλο αριθμό νευρώνων να έχουμε υπερεκπαίδευση. Όταν το πλήθος των δεδομένων εισόδου είναι ίσο με 12,384, ένα στρώμα με έξοδο 10,000 νευρώνες είναι πιθανό να «κρατάει ίδιο» μεγάλο μέρος της εισόδου και πρακτικά να μην εκπαιδεύει σχεδόν καθόλου. Για το λόγο αυτό ένα ποσοστό 25-35% του αριθμού των δεδομένων εισόδου είναι μια σχετικά καλή επιλογή νευρώνων εξόδου για το RBF layer.

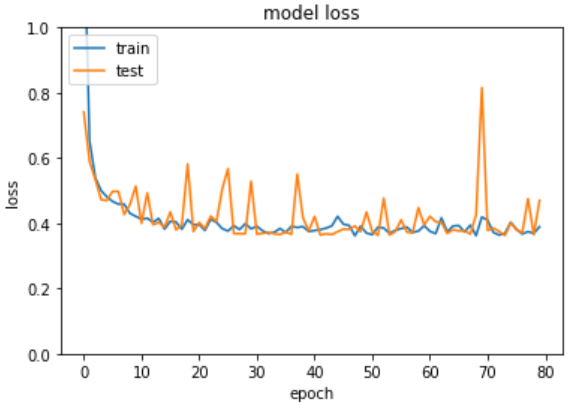
Στη συνέχεια, η παράμετρος που μελετήθηκε στο RBF neural network μας είναι το gamma. Το gamma σαν ορισμός δείχνει την μεγαλύτερη ή μικρότερη "επιρροή" ενός στοιχείου στη διαδικασία του training (με μικρό gamma να δείχνει μεγάλη επιρροή και αντίστοιχα μεγάλο gamma, μικρότερη). Η επιρροή αυτή μπορεί να ερμηνευθεί χρησιμοποιώντας και το μέγεθος της ακτίνας των «σχημάτων» που δημιουργούνται με μία RBF, εννοώντας πως μεγάλο gamma προσδιορίζει μικρή ακτίνα και άρα αυστηρότερη κατάταξη των δεδομένων, και το αντίστροφο. Ένα πολύ μεγάλο gamma είναι φυσικά προβληματικό, καθώς μπορεί να οδηγήσει σε ακτίνες που περιέχουν ουσιαστικά μόνο ένα στοιχείο η καθεμία.

Gamma	Neurons	Initializer	Optimizer	RMSE	R^2	Time (s)
0.001	4000	KMeans	Adam	0.725384825	0.4738168545	271.0691
0.01	4000	KMeans	Adam	0.685614676	0.5299325146	261.4467
0.1	4000	KMeans	Adam	0.607473289	0.6309762024	274.4321
1	4000	KMeans	Adam	0.502741119	0.7472513672	275.5470
10	4000	KMeans	Adam	0.751252907	0.4356190685	275.5609
100	4000	KMeans	Adam	0.999994846	0.0000103062	262.6518

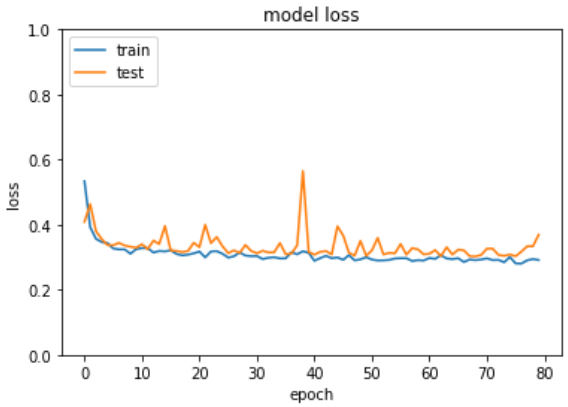
a) Gamma = 0.001



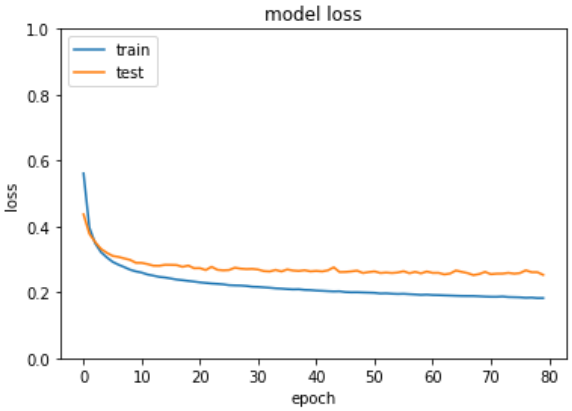
b) Gamma = 0.01



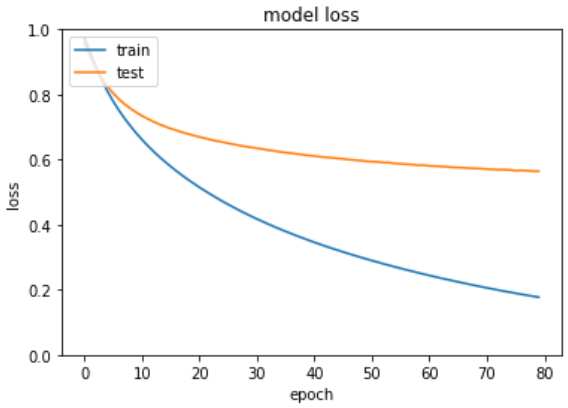
c) Gamma = 0.1



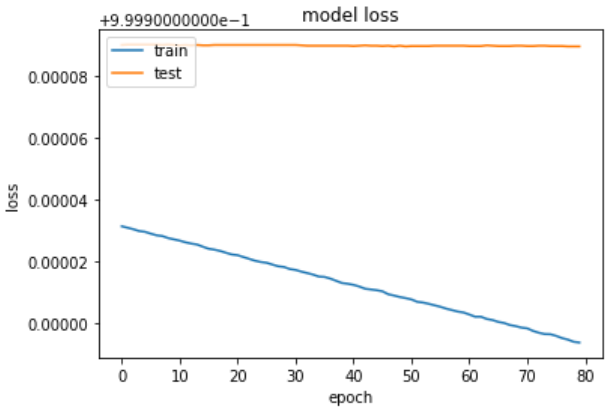
d) Gamma = 1



e) Gamma = 10



f) Gamma = 100



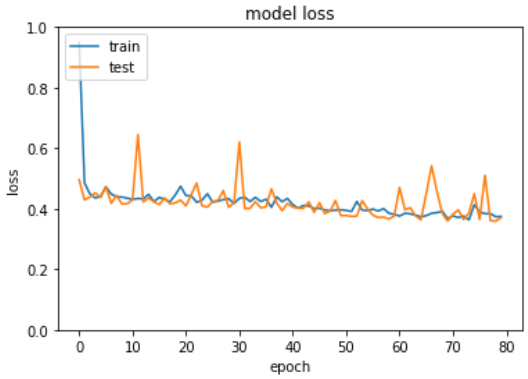
Αρχικά, σχολιάζεται πως ο χρόνος δεν εξαρτάται ιδιαίτερα από το gamma, καθώς υπάρχουν μόνο μικρές αμελητέες μεταβολές του. Το καλύτερο μοντέλο εδώ φαίνεται να εκπαιδεύεται για gamma = 1, κάτι σχετικά αναμενόμενο καθώς είναι μια μεσαία τιμή. Για αυτό το gamma παρουσιάζεται και το καλύτερο R^2 που έχουμε δει μέχρι στιγμής, που αγγίζει το 75%. Όπως αναφέρθηκε και πιο πάνω, τα μεγάλα γ οδηγούν σε πολύ κακών αποτελεσμάτων μοντέλα, όπως φαίνεται στις τιμές των R^2, RMSE αλλά και στα σχήματα.

Η επόμενη αλλαγή παραμέτρου στην οποία προχωρήσαμε είναι η αλλαγή του initializer/αρχικοποιητή των βαρών του RBF layer. Εδώ σημειώνεται πως δημιουργήθηκε και μία συνάρτηση για αρχικοποίηση των βαρών με τον αλγόριθμο KMeans, ο οποίος δέχεται σαν είσοδο το σύνολο εκπαίδευσης. Είναι γνωστό πως η αρχικοποίηση βαρών με KMeans είναι αρκετά χρήσιμη και συχνά οδηγεί σε καλά αποτελέσματα.

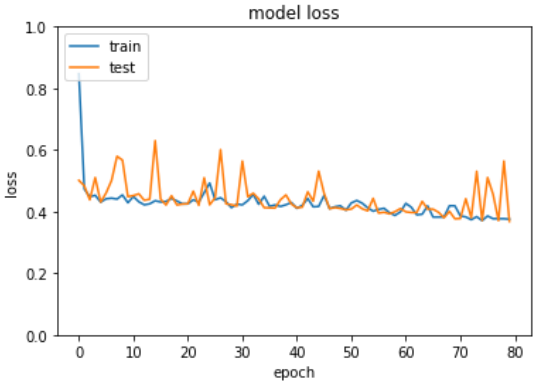
Initializer	Neurons	Gamma	Optimizer	RMSE	R^2	Time (s)
Uniform	4000	0.1	Adam	0.6087319704	0.62944538813	190.1418
Zeros	4000	0.1	Adam	0.6066614363	0.63196190166	189.1458
Random normal	4000	0.1	Adam	0.6099252035	0.62799124607	188.9637
He normal	4000	0.1	Adam	0.5856297440	0.65703780286	189.2821
Glorot normal	4000	0.1	Adam	0.6289332875	0.60444291976	190.1398
Kmeans	4000	0.1	Adam	0.5541909893	0.69287234735	268.1642

Σχολιάζεται εδώ πως ο KMeans δίνει τα καλύτερα αποτελέσματα, χωρίς να έχει όμως μεγάλη διαφορά από τους άλλους initializers. Φυσικά, στην περίπτωση του KMeans έχουμε μεγαλύτερο χρόνο, καθώς ο αλγόριθμος πρέπει να κάνει adapt τα βάρη στο σύνολο εισόδου. Παρακάτω παρουσιάζονται τα διαγράμματα σφάλματος στο σύνολο εκπαίδευσης και ελέγχου για τον παραπάνω πίνακα.

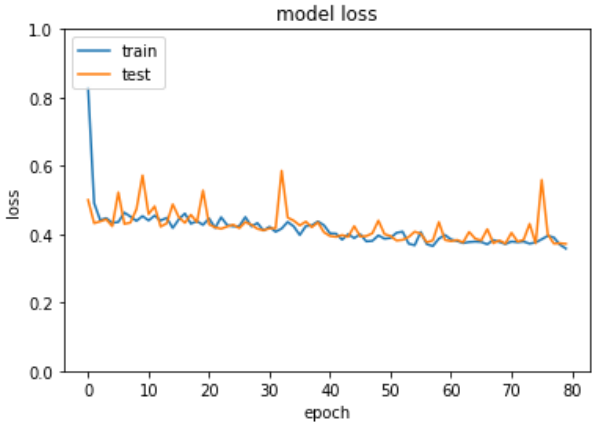
a) Uniform



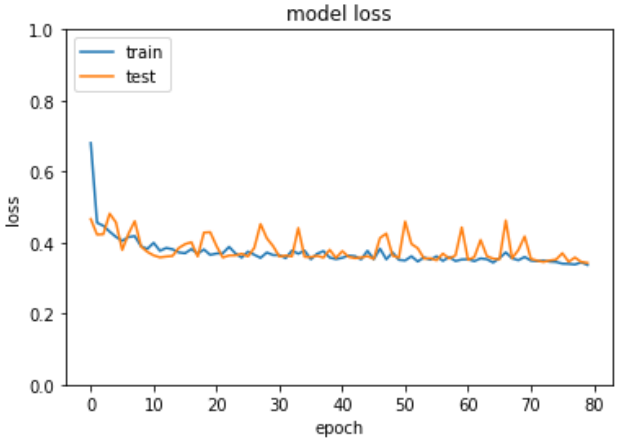
b) Zeros



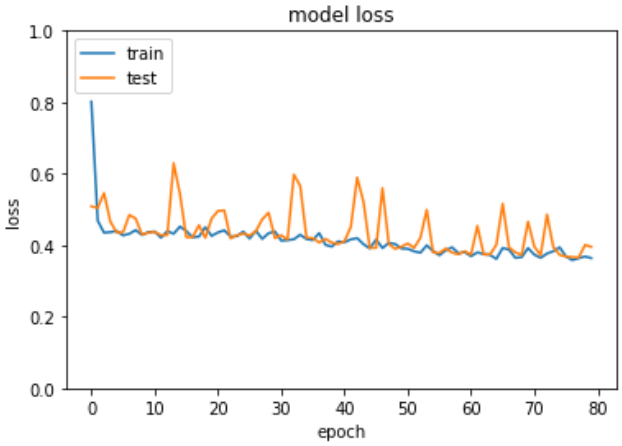
c) Random normal



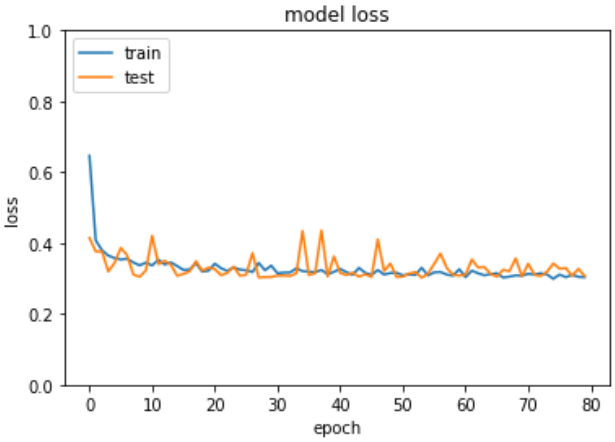
d) He normal



e) Glorot normal



f) Kmeans



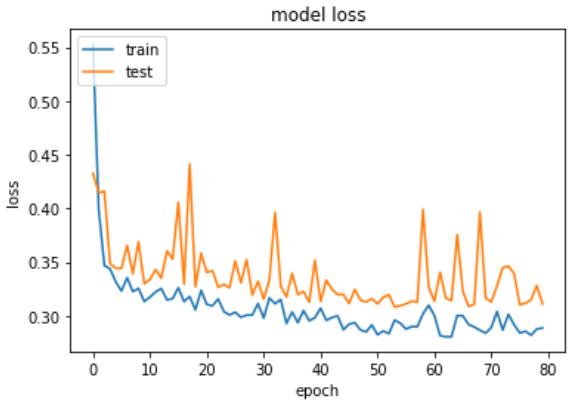
Τελευταία παράμετρος που «πειράχθηκε» στη μελέτη μας είναι ο optimizer στην εκπαίδευση του μοντέλου. Όπως είδαμε και παραπάνω, αυτός που χρησιμοποιήθηκε κατεξοχήν είναι ο Adam (με default learning rate = 0.001), ο οποίος συνδυάζει θεωρητικά τα καλύτερα χαρακτηριστικά των κλασσικών optimizer (π.χ. SGD και RMSprop). Στον παρακάτω πίνακα παρουσιάζονται τα αποτελέσματα των πειραμάτων όσον αφορά διαφορετικούς optimizer.

Optimizer	Neurons	Gamma	Initializer	RMSE	R^2	Time (s)
Adam lr=0.001	4000	0.1	KMeans	0.5578920836	0.68875642303	238.6215
SGD lr=0.01	4000	0.1	KMeans	0.9384818590	0.11925180030	237.4506
SGD lr=0.1	4000	0.1	KMeans	0.6825969508	0.53406140263	267.5123
RMSprop lr=0.001	4000	0.1	KMeans	0.8429908396	0.28936644431	243.2992
RMSprop lr=0.1	4000	0.1	KMeans	0.6168564544	0.61948811463	261.4333
Adadelta lr=0.001	4000	0.1	KMeans	0.7117490239	0.49341332686	243.9120
Adadelta lr=1	4000	0.1	KMeans	0.5337863408	0.71507214231	233.2708

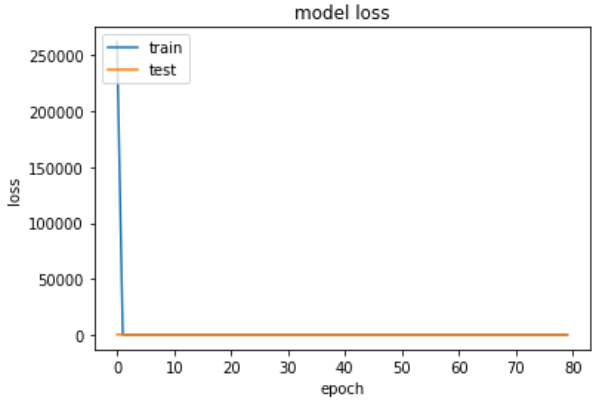
Την καλύτερη συμπεριφορά φαίνεται να την παρουσιάζει ο optimizer με υλοποίηση του αλγορίθμου Adadelta και learning rate = 1 (στη βιβλιογραφία αναφέρεται πως αυτή είναι η τιμή που πρώτα μελετήθηκε για τον αλγόριθμο και παρουσιάζει πιθανώς καλά αποτελέσματα). Στη συνέχεια ακολουθεί ο αλγόριθμος Adam, αυτός δηλαδή που χρησιμοποιήσαμε στα άλλα πειράματα. Φαίνεται γενικότερα (αν εξαιρέσουμε τον Adam) πως μεγαλύτερο learning rate στο πρόβλημα μας δίνει και καλύτερα αποτελέσματα, ίσως λόγω των σχετικά λίγων epochs.

Επίσης, σε κάποιες από τις επαναλήψεις των παραπάνω πειραμάτων εμφανίστηκε υπερεκπαίδευση, δε παρουσιάζονται όμως εδώ. Παρακάτω παρατίθενται τα διαγράμματα σφάλματος για μια τελευταία φορά, με βάση την αλλαγή των optimizer.

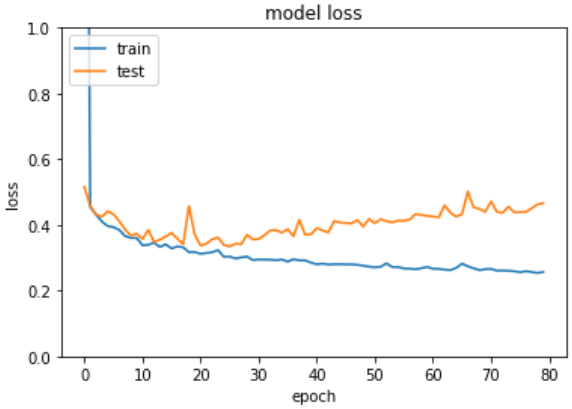
a) Adam (lr = 0.001)



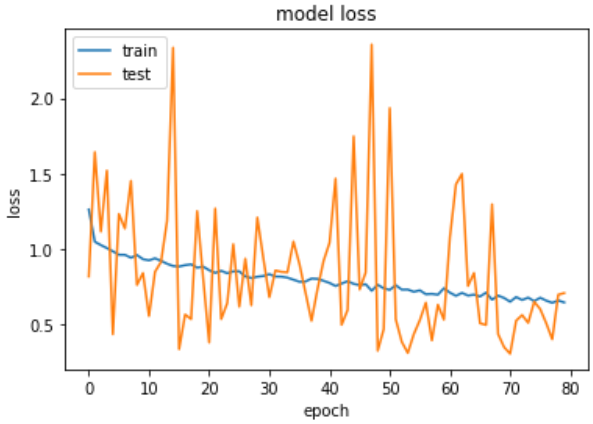
b) SGD (lr = 0.01)



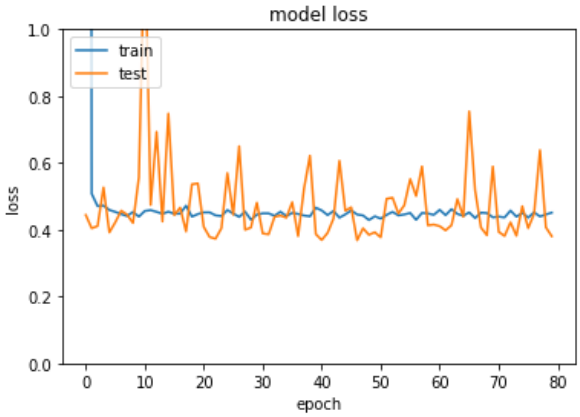
c) SGD (lr = 0.1)



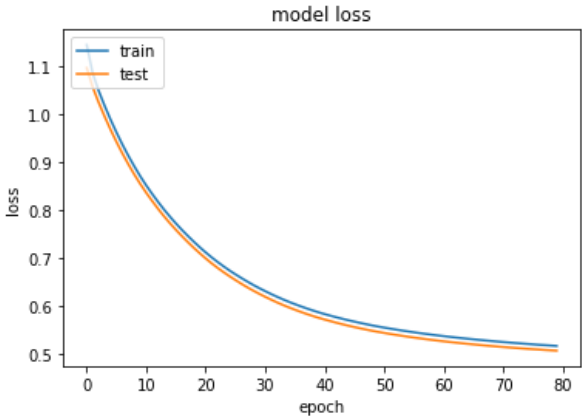
d) RMSprop (lr = 0.001)



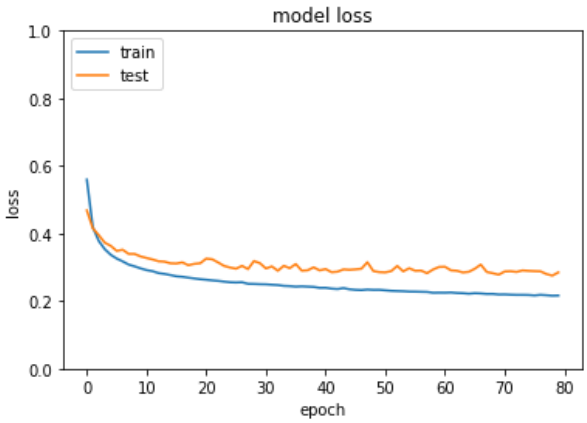
e) RMSprop (lr = 0.1)



f) Adadelta (lr = 0.001)



g) Adadelta (lr = 1)



Σχολιάζεται πως ο RMSprop έχει πολύ έντονες αυξομειώσεις και διακυμάνσεις, ειδικά στο σφάλμα του συνόλου ελέγχου (όπως και ο Adam αλλά σε μικρότερο βαθμό). Αυτό θα μπορούσε να δημιουργήσει προβλήματα αν επιλεγόταν ο συγκεκριμένος optimizer σε μια πραγματική υλοποίηση. Σημειώνεται, τέλος, πως ο SGD με learning rate = 0.01 έχει μια τάση υπερεκπαίδευσης.

Σύγκριση με τον αλγόριθμο k-nearest neighbors (με διάφορα k)

Σαν τελευταίο κομμάτι της εργασίας, παρουσιάζονται παρακάτω τα αποτελέσματα του αλγορίθμου k-nearest neighbors (για k = 1, k = 3 & k = 5), στο ίδιο πρόβλημα regression που μελετήθηκε παραπάνω, χάριν σύγκρισης.

k	RMSE	R^2	Time (s)
1	0.700360624748365	0.509494995302079	0.260057210922241
3	0.578277969669901	0.665594589794456	0.378086566925048
5	0.569451603686354	0.675724871059039	0.418094873428344

Τα αποτελέσματα είναι πολύ ενθαρρυντικά, ειδικά για k neighbors = 5. Οι χρόνοι από την άλλη είναι πολύ πολύ μικρότεροι σε σχέση με το RBF neural network που δημιουργήσαμε παραπάνω, οπότε σίγουρα ο αλγόριθμος knn θα ήταν μια πολύ καλή εναλλακτική για τη μελέτη του συγκεκριμένου προβλήματος.

ΒΙΒΛΙΟΓΡΑΦΙΑ

https://github.com/PetraVidnerova/rbf_keras

https://www.kaggle.com/residentmario/radial-basis-networks-and-custom-keras-layers

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html

https://www.tutorialspoint.com/keras/keras_customized_layer.htm

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

https://towardsdatascience.com/radial-basis-function-neural-network-simplified-6f26e3d5e04d

https://www.sciencedirect.com/topics/veterinary-science-and-veterinary-medicine/radial-basis-function-network

https://numpy.org/doc/stable/user/basics.broadcasting.html

https://www.youtube.com/watch?v=1Cw45yNm6VA&ab_channel=macheads101

https://www.statology.org/how-to-interpret-rmse/

https://inria.github.io/scikit-learn-mooc/python_scripts/datasets_california_housing.html

http://docplayer.gr/63321421-Diktya-synartiseon-vasis-aktinikoy-typoy-radial-basis-functions-rbf.html

https://keras.io/api/optimizers/