**ΠΡΩΤΗ ΕΡΓΑΣΙΑ: MNIST Classification MultiLayer Perceptron**

**ΜΑΘΗΜΑ: ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ - ΒΑΘΙΑ ΜΑΘΗΣΗ**

**ΓΙΑΝΝΑΚΟΠΟΥΛΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ , ΑΕΜ: 16051** (τμήμα Μαθηματικών)

**ΕΙΣΑΓΩΓΗ**

**Βάση Δεδομένων:**

Για την παρούσα εργασία, καθώς και για την επικείμενη 1η Εργασία, χρησιμοποιήθηκε η βάση δεδομένων MNIST, μια πολύ γνώστη βάση δεδομένων, υποσύνολο της μεγαλύτερης NIST. Περιέχει χειρόγραφα ψηφία αριθμών 0,1,…,9. Στην ιστοσελίδα http://yann.lecun.com/exdb/mnist/ υπάρχουν τέσσερα αρχεία διαθέσιμα: ένα training set 60,000 παρατηρήσεων και τα αντίστοιχα labels, καθώς και ένα test set 10,000 παρατηρήσεων και τα αντίστοιχα labels. Οι εικόνες που περιέχονται σε αυτά είναι διαστάσεων 28x28 σε επίπεδα του γκρι και έχει κανονικοποιηθεί το μέγεθος τους αλλά και έχουν κεντροποιηθεί.

**Γλώσσα Προγραμματισμού:**

Χρησιμοποιήθηκε η -ανοικτού κώδικα- γλώσσα προγραμματισμού R, που χρησιμοποιείται ευρέως στον τομέα της υπολογιστικής στατιστικής. Με τις κατάλληλες βιβλιοθήκες και με τη νεοσύστατη βιβλιοθήκη Keras, δίνεται η δυνατότητα υλοποίησης πολλών αλγορίθμων, όπως o kNN και ο nearest Centroid που χρησιμοποιήθηκαν εδώ, αλλά και άλλων αλγορίθμων στατιστικής μάθησης αλλά και αρχιτεκτονικών βαθιάς μάθησης.

**Preprocessing**

Στη βιβλιοθήκη Keras της R, περιέχονται μερικά από τα πιο γνωστά dataset, όπως η MNIST, Cifar100, Cifar10 κλπ. Αφού, λοιπόν, φορτώθηκαν τα δεδομένα και ονομάστηκαν ως:

x\_train: 60,000 x 28 x 28

y\_train: 60,000 (τα αντίστοιχα labels)

x\_test: 10,000 x 28 x 28

y\_test: 10,000 (τα αντίστοιχα labels)

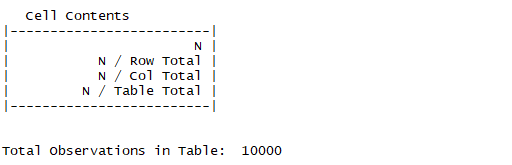
Στη συνέχεια, οι 28x28 πίνακες των x\_train και x\_test μετατράπηκαν σε διανύσματα 784 θέσεων. Έπειτα έγινε κανονικοποίση των τιμών τους στο διάστημα [0,1].   
Τα y\_train και y\_test θα μετατραπούν σε δυαδικά διανύσματα 10 θέσεων (one-hot encoding) μόνο για τα Νευρωνικά δίκτυα.

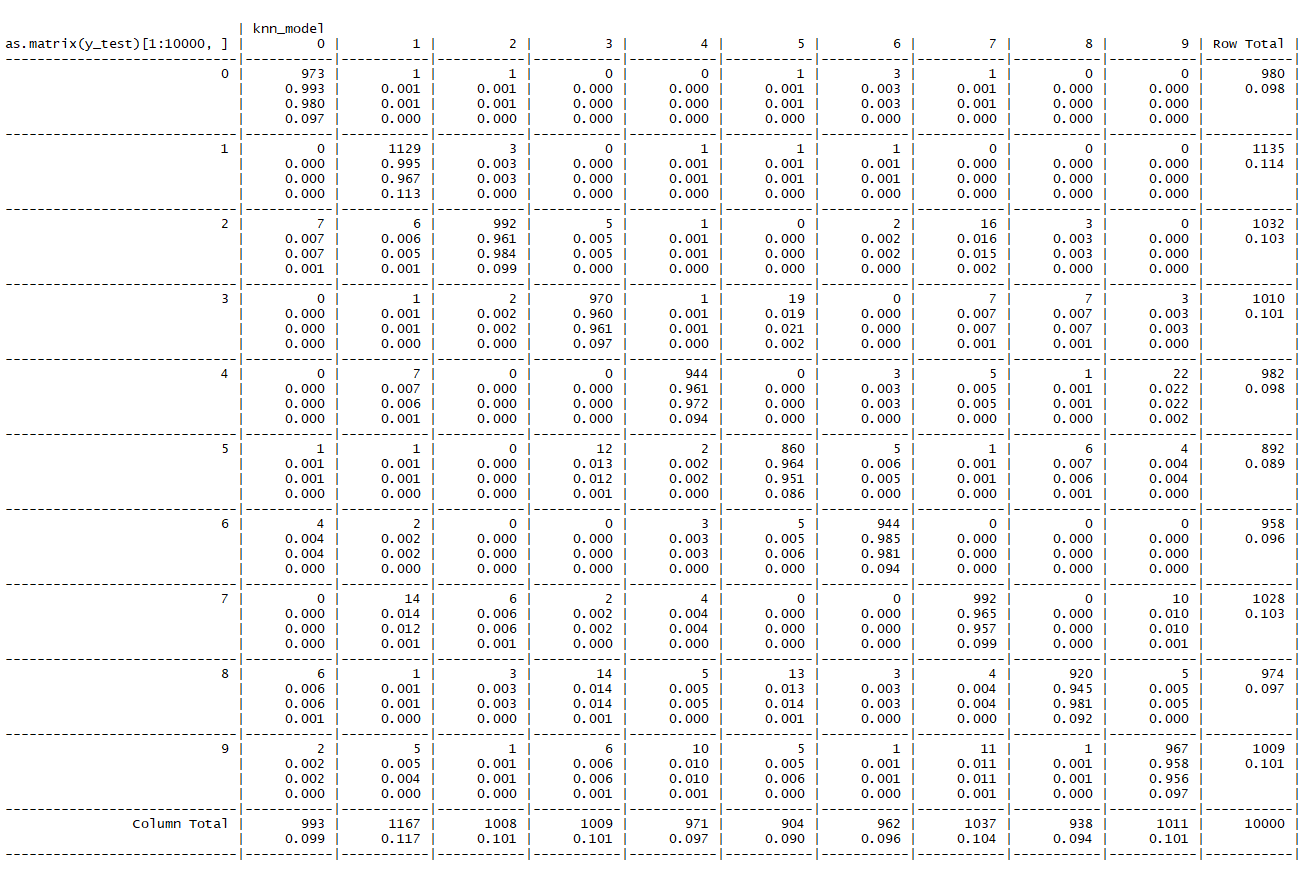
**k-nearest neighbors algorithm**

Ο αλγόριθμος kNN λαμβάνει κάθε παρατήρηση του test set και υπολογίζει την Ευκλείδεια απόσταση της από κάθε παρατήρηση του training set. Εν συνεχεία, επιλέγονται και οι k μικρότερες αποστάσεις, δηλαδή οι k κοντινότερες παρατηρήσεις του training set. Τέλος, αυτές "ψηφίζουν" με βάση το label τους και η πλειοψηφία δίνεται ως απάντηση στην ταξινόμηση της παρατήρησης του test set.

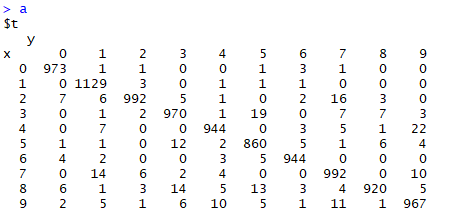
Ο αλγόριθμος k - κοντινότεροι γείτονες εφαρμόσθηκε για k=1 και k=3. Η απόσταση που χρησιμοποιήθηκε ήταν η Ευκλείδεια, ενώ σε περίπτωση ισοψηφίας λαμβάνει υπόψη την "ψήφο" όλων όσων απέχουν ίση απόσταση με την k-ιοστή.

* Για την πρώτη περίπτωση το Classification του αλγορίθμου για το test set φαίνεται στον παρακάτω πίνακα σύγχυσης.





Ή απλούστερα στον παρακάτω:

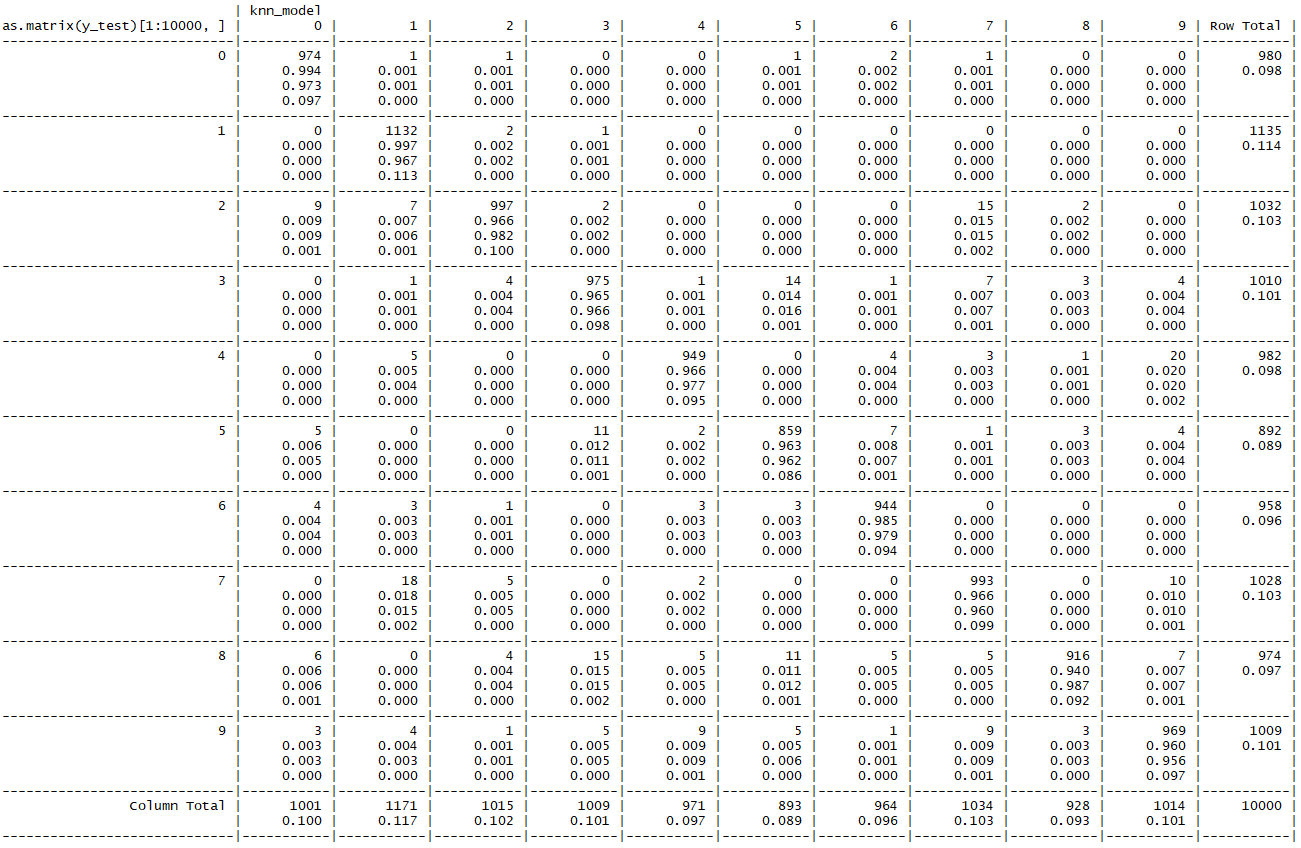


Ως μετρική αξιολόγησης έχουμε το accuracy= (TP+TN)/(TP+TN+FP+FN):

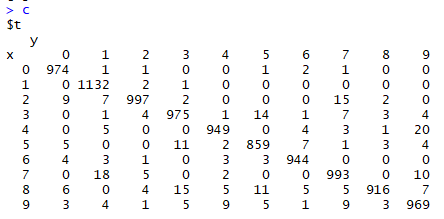
(περισσότερες μετρικές θα χρησιμοποιηθούν στην 1η εργασία)



Επομένως, ο αλγόριθμος kNN με 1 κοντινότερο γείτονα έδωσε ακρίβεια 96.91% ή αλλιώς το test error rate ήταν 3.09% στις 10,000 παρατηρήσεις του test set.

* Στη συνέχεια, ο αλγόριθμος για k=3 έδωσε τον παρακάτω πίνακα σύγχυσης:

Ή καλύτερα:

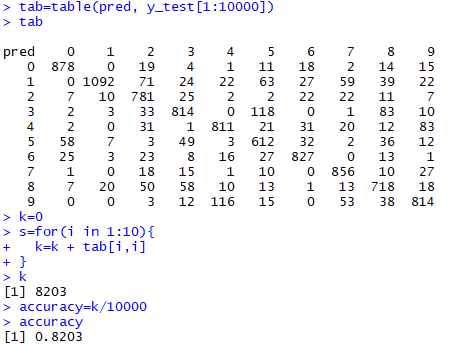


Ο αλγόριθμος kNN με k=3, έδωσε ακρίβεια 97.08%, ή test error rate 2.92% στις 10,000 παρατηρήσεις του test set.



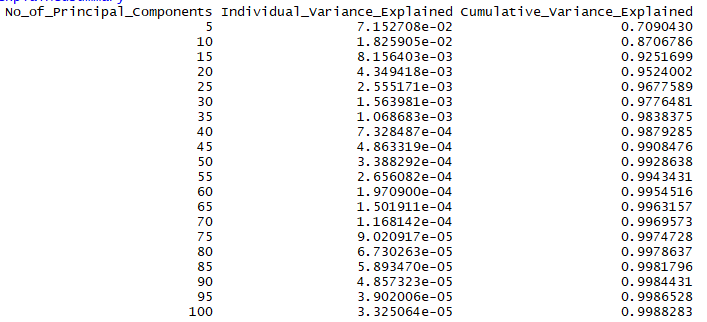
**Nearest centroid classifier**

Ο αλγόριθμος κοντινότερου κέντρου εφαρμόζεται παίρνοντας τη μέση τιμή των παρατηρήσεων του training set ως κέντρο και στη συνέχεια μετρώντας την εγγύτητα κάθε παρατήρησης του test set από αυτά. Το κέντρο που απέχει τη μικρότερη απόσταση δίνει και το label της κλάσης του στην παρατήρηση του test set. Ο αλγόριθμος εφαρμόσθηκε στα ίδια δεδομένα, με την Ευκλείδεια απόσταση. Τα αποτελέσματα της κατηγοριοποίησης φαίνονται στον παρακάτω πίνακα σύγχυσης:



Για τον έλεγχο της κατηγοριοποίησης χρησιμοποιήθηκε η μετρική accuracy, με αποτέλεσμα 82.03% ακρίβεια, δηλαδή με test error rate 17.97% στις 10,000 παρατηρήσεις του test set.

**Multilayer Perceptron**

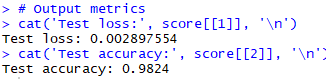
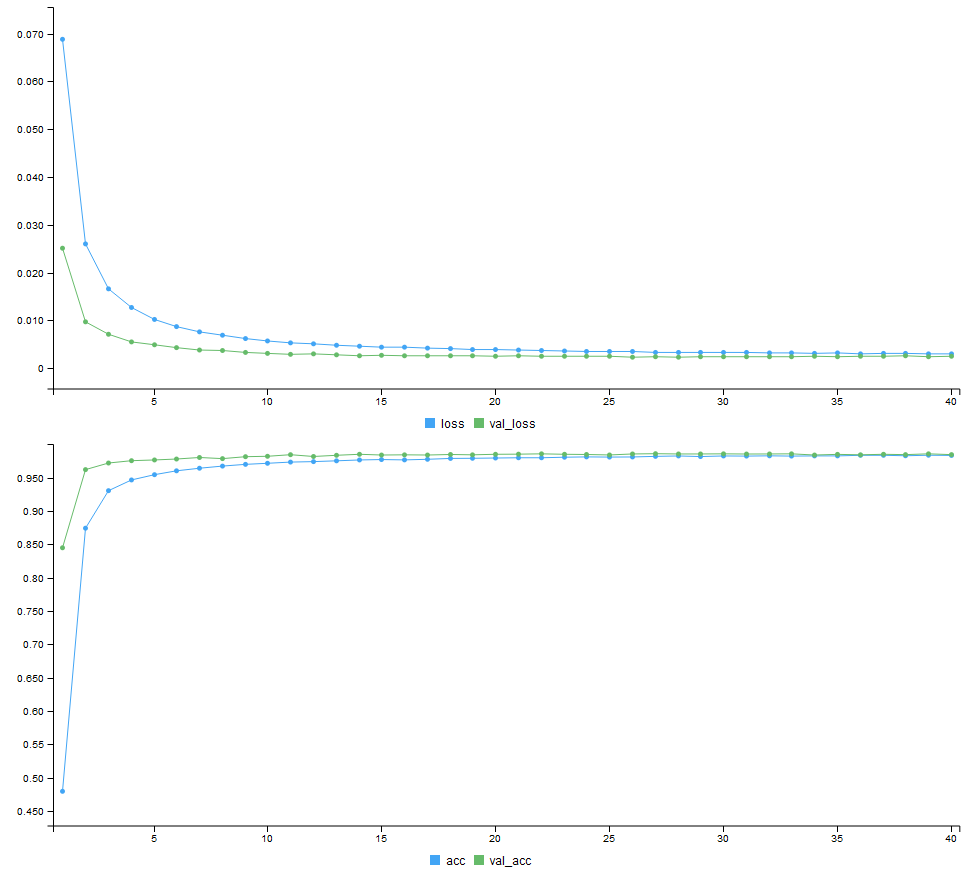
Αρχικά, τα δεδομένα χωρίστηκαν σε x\_train (60,000x28x28), y\_train(60,000x1), x\_test (10,000x28x28) και y\_test(10,000x1). Τα x\_train και y\_train μετατράπηκαν σε 60,000x784 και 10,000x784 αντίστοιχα ώστε κάθε γραμμή να περιγράφει μια εικόνα (input). Στη συνέχεια, αφού κάθε τιμή ανήκει στο [0,255] έγινε scaling στο διάστημα [0,1]. Τέλος, χρησιμοποιήθηκε η μέθοδος PCA, προκειμένου να μειωθεί η διάσταση των δεδομένων, από τις 784 διαστάσεις στις οποίες βρισκόμαστε. Τα αποτελέσματα της PCA φαίνονται παρακάτω:

Το 99.54% της διακύμανσης των δεδομένων απεικονίζεται στις πρώτες 60 κύριες συνιστώσες. Επομένως, η διάσταση των δεδομένων μειώνεται αισθητά.

Από τη βιβλιοθήκη Keras, επιλέχθηκε το μοντέλο sequential και μένει να προσδιοριστούν οι παράμετροι του.

Αρχικά, το νευρωνικό δίκτυο θα αποτελείται από 4 layers. Στο πρώτο θα δίνεται ως input ένα διάνυσμα 60 θέσεων και το output θα είναι διάστασης 256, όση και η διάσταση του πρώτου κρυφού layer. Η συνάρτηση ενεργοποίησης είναι η ReLU. Το dropout rate έχει οριστεί στο 0.2. Στο δεύτερο layer έχουμε και πάλι ως συνάρτηση ενεργοποίησης τη ReLU και το output είναι 128 διαστάσεων. Το dropout rate και πάλι 0.2. Το τελευταίο layer έχει επίσης ως συνάρτηση ενεργοποίησης τη ReLU και output 10, όσες και οι κλάσεις των labels. Στο compile, ως συνάρτηση κόστους χρησιμοποιήθηκε το μέσο τετραγωνικό σφάλμα (MSE), ως optimizer το rmsprop και ως μετρική το accuracy. Τέλος, το μοντέλο κάνει train στα δεδομένα, παίρνοντας batches διάστασης 256 σε κάθε μια από τις 40 εποχές, ενώ γίνεται 10-fold cross-validation.

Κάθε εποχή διαρκεί περίπου 1 δευτερόλεπτο και τα αποτελέσματα του μοντέλου στο training και στα άγνωστα δεδομένα του test set είναι:

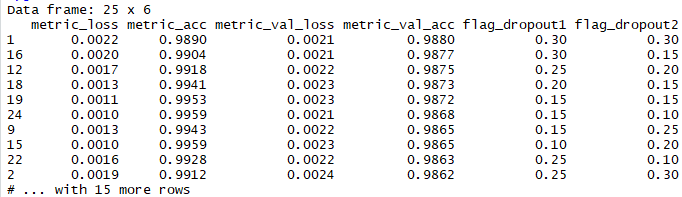
Το validation accuracy είναι 98.48%.

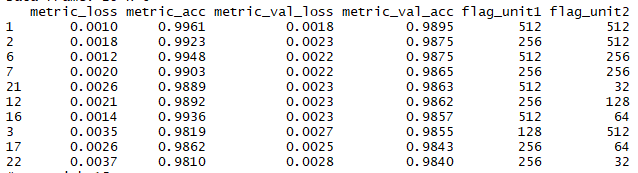
Έχουμε λοιπόν accuracy στο test set 98.24%.

* Εν συνεχεία, προκειμένου να βρεθούν οι βέλτιστες υπερπαράμετροι, χρησιμοποιήθηκε η συνάρτηση tuning\_run από τη βιβλιοθήκη tfruns.

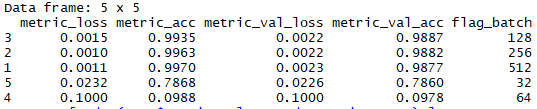
Αρχικά για να βρεθούν οι βέλτιστες τιμές για τα dropout rates δοκιμάστηκαν όλοι οι συνδυασμοί (25) για τις τιμές 0.1, 0.15, 0.2, 0.25 και 0.3.

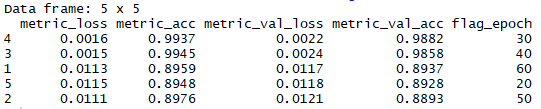
Παρακάτω φαίνεται ότι την καλύτερη επίδοση στο validation accuracy την έδωσαν οι τιμές 0.3 τόσο για το dropout1 όσο και για το dropout2.



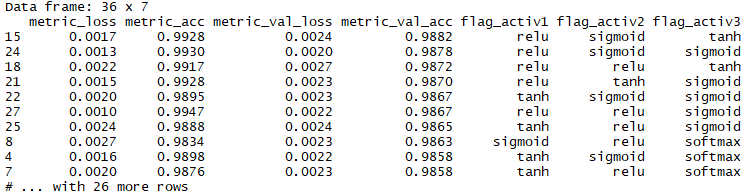
Σταθεροποιώντας τα dropout rates στο 0.3, δοκιμάζουμε τις τιμές 32, 64, 128, 256, 512 σε όλους τους πιθανούς συνδυασμούς για τον αριθμό των νευρώνων στα hidden layers.

Παρατηρούμε ότι το καλύτερο accuracy στο validation set το δίνουν οι τιμές 512 και για τα δυο layers.

Έχοντας επομένως τα dropout rates στο 0.3 και τον αριθμό των νευρώνων στα κρυφά layers στους 512, δοκιμάζουμε διάφορες τιμές για το batch size: 32, 64, 128, 256, 512.

Κρατάμε, λοιπόν το batch size στο 128. Στη συνέχεια, δοκιμάζουμε διάφορες τιμές για τις εποχές: 20, 30, 40, 50, 60.

Καλύτερο validation accuracy δίνουν οι 30 εποχές.

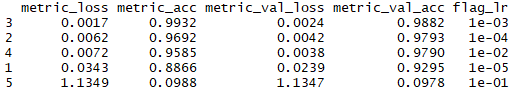
Έπειτα, θα δοκιμάσουμε διάφορους συνδυασμούς συναρτήσεων ενεργοποίησης για τα layers:

Ο καλύτερος συνδυασμός είναι:

1ο layer: ReLU

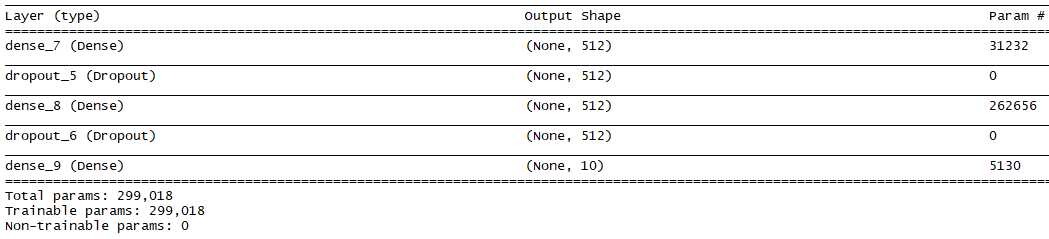
2o layer: Sigmoid

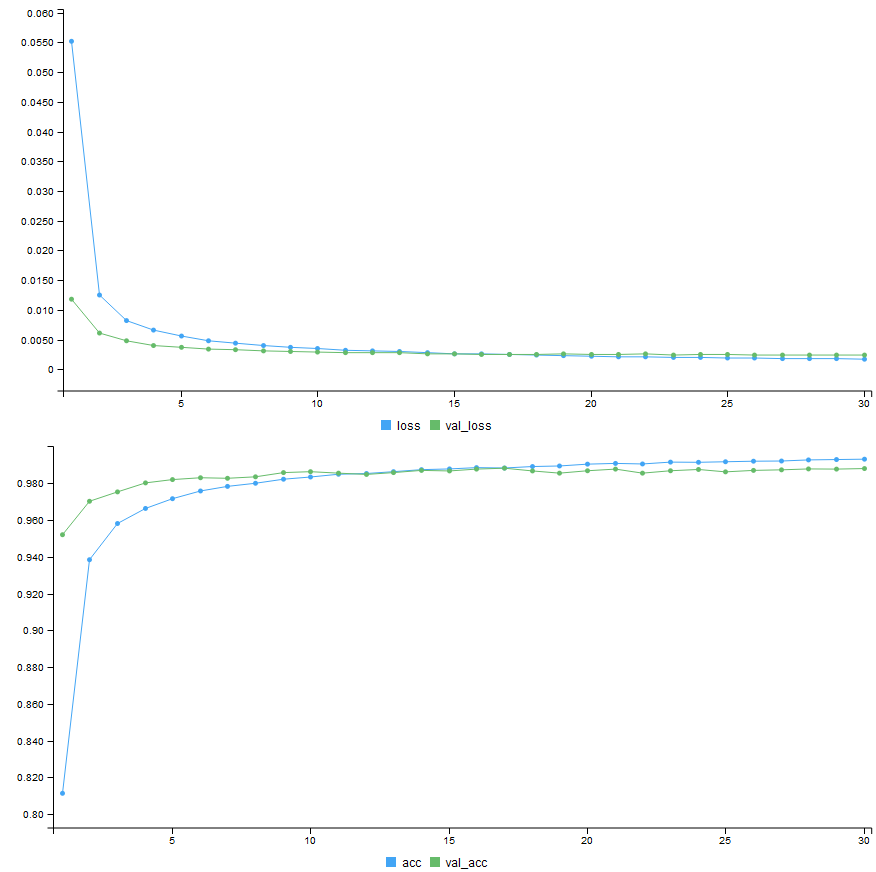
3o layer: tanh

Για τον optimizer rsmprop, γίνεται έλεγχος για διάφορες τιμές του learning rate: 0.1, 0.01, 0.001, 0.0001, 0.00001

Βέλτιστο learning rate φαίνεται να είναι το 0.001.

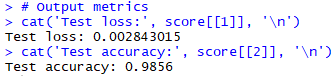
Εκπαιδεύοντας το νευρωνικό με δύο hidden layers, συναρτήσεις ενεργοποίησης ReLU, sigmoid, tanh αντίστοιχα, 512 νευρώνες σε κάθε κρυφό επίπεδο, dropout rates 0.3, optimizer rmsprop με learning rate 0.001, συνάρτηση κόστους MSE και μετρική το accuracy, 30 εποχές (4s/epoch), batch size 128 και 10-fold c-v στο fit, έχουμε τα εξής αποτελέσματα:





.

Το validation accuracy είναι 98.8%, ενώ το accuracy στο test set είναι 98.56%, βελτιωμένο σε σχέση με το αρχικό κατά 0.32% (32 παραπάνω σωστές κατηγοριοποιήσεις).



Σε αυτό το σημείο μπορούμε να συγκρίνουμε με τα ανάλογα αποτελέσματα των αλγορίθμων k nearest neighbours και nearest centroid που έδωσαν αποτελέσματα 96.91% (k=1), 97.08% (k=3) και 82.03% στα ίδια δεδομένα.

Τέλος, το accuracy φτάνει το 98.68%, βελτιωμένο σε σχέση με το προηγούμενο κατά 0.12% αν προστεθεί άλλο ένα hidden layer με 512 νευρώνες και activation ReLU, όπως φαίνεται παρακάτω:

