Τρίτη εργασία

Ματάκος Αλέξανδρος

Εισαγωγή

Για την εργασία επιλέχθηκε το πρόβλημα της αναγνώρισης χειρόγραφων ψηφίων της βάσης δεδομένων **MNIST**.
Το πρόγραμμα είναι γραμμένο σε **Python** και υλοποιεί ένα **Radial Basis Function Neural Network**, το οποίο πραγματοποιεί multi-class classification στις δέκα κλάσεις της **MNIST**.

Τα δεδομένα

Η βάση δεδομένων αποτελείται απο 50.000 training samples και 10.000 test samples. Τα δείγματα είναι σε μορφή (784,1) arrays, με τιμές οι οποίες είναι scaled down σε 0-1.
Το πρόγραμμα χρησιμοποιεί τα training_data και test_data. Το training_data είναι ένα tuple που αποτελείται από δύο εισόδους. Η πρώτη είσοδος είναι οι 50000 φωτογραφίες των χειρόγραφων ψηφίων σε μορφή (50000, 784) numpy array, ενώ η δεύτερη είσοδος είναι το ψηφίο που αντιστοιχεί στην κάθε εικόνα σε μορφή (50000,) numpy array.
Τα test_data είναι της ίδιας μορφής, όμως με 10000 δείγματα.

Πρόγραμμα

Το RBF Neural Network αποτελείται από τρία layers: input layer, hidden layer και output layer.
Οι νευρώνες του κρυφού στρώματος ειναι τα 'radial units' και έχουν ως συνάρτηση ενεργοποίησης κάποια ακτινική συνάρτηση (radial function). Το πρόγραμμα δέχεται κάποια είσοδο σε μορφή διανύσματος-στήλης και υπολογίζεται μέσω του κάθε νευρώνα του κρυφού δικτύου η απόσταση της κάθε εισόδου απο κάθε κέντρο (το οποίο αντιστοιχεί σε κάθε νευρώνα). Έπειτα υπολογίζεται το σταθμισμένο άθροισμα των ενεργοποιήσεων των νευρώνων του κρυφού επιπέδου, και συγκρίνεται με το desired output, το οποίο είναι το label που αντιστοιχεί σε κάθε είσοδο.

Το κρυφό στρώμα είναι συνδεδεμένο με το output layer μέσω των βαρών w_{ji}, τα οποία συμβολίζουν το βάρος που συνδέει τον i-οστό νευρώνα του κρυφού επιπέδου με τον j-οστό νευρώνα του output layer.
Το output layer αποτελείται απο δέκα νευρώνες (έναν για κάθε κλάση) με γραμμική συνάρτηση ενεργοποίησης για τον j-οστό νευρώνα:

$$f_j(x) = \sum_{i=1}^{k} w_{ji} \cdot \varphi(\|x_i - c_i\|)$$

όπου φ είναι η ακτινική συνάρτηση, x_i το input vector, c_i το κέντρο που αντιστοιχεί στον i-οστό νευρώνα του κρυφού επιπέδου και k το μέγεθος του κρυφού επιπέδου (ή το πλήθος των κέντρων αντίστοιχα).

Το πρόγραμμα υπολογίζει την απόσταση του κάθε στοιχείου του training data set από όλα τα κέντρα c_i και αποθηκεύει τις αποστάσεις αυτές σε έναν πίνακα $\mathbf{G} = (g_{ji})$, όπου g_{ji} είναι η απόσταση του j-οστού training data sample από το i-οστό κέντρο. Έπειτα αναζητεί έναν πίνακα $\mathbf{W} = (w_{ji})$ ώστε το σύστημα

$$\mathbf{G} \cdot \mathbf{W} = \mathbf{D}$$

να έχει βέλτιστη λύση, όπου D ο πίνακας που περιέχει ως γραμμές τα desired outputs σε μορφή διανυσμάτων. Ο πίνακας $\mathbf{D}$ έχει μορφή 50.000×10, μία γραμμή για κάθε data point, και κάθε γραμμή του αποτελείται από μηδενικά εκτός από έναν άσσο στη θέση που αντιστοιχεί στο ψηφίο του label της εισόδου. Αυτό γίνεται για να μπορεί να συγκριθεί η κάθε γραμμή του $\mathbf{D}$ με το output του output layer.
Με βάση τη θεωρία, όπου τα $\mathbf{W}$, $\mathbf{D}$ είναι τα διανύσματα w, d λύνω το σύστημα με τη μέθοδο ελαχίστων τετραγώνων (Least Mean Squares). Η λύση δίνεται με χρήση του Moore-Penrose ψευδοαντίστροφου και έχει τη μορφη:

$$\mathbf{W} = G^{\dagger} \cdot \mathbf{D}$$

όπου $G^{\dagger}$ ο Moore-Penrose ψευδοαντίστροφος.
Η εκπαίδευση ολοκληρώνεται με τον υπολογισμό του πίνακα βαρών $\mathbf{W}$.
Έπειτα εκτελούμε την πράξη $\mathbf{G} \cdot \mathbf{W}$ και συγκρίνουμε τον πίνακα που προκύπτει με τον πίνακα $\mathbf{D}$, ώστε να συγκρίνουμε τα αποτελέσματα και να βρούμε την απόδοση στο train data set.
Έπειτα ξαναυπολογίζεται ο πίνακας $\mathbf{G}$ για το test data set και πολλαπλασιάζεται απο αριστερά με τον $\mathbf{W}$. Ο πίνακας που προκύπτει συγκρίνεται πάλι με τον $\mathbf{D}$, αυτή τη φορά του test data set, ώστε να βρεθεί η απόδοση του δικτύου στο test data set.

Το πρόγραμμα έχει τις εξής παραμέτρους:

- **Radial Function**. Μπορεί να είναι μία από τις εξής:

  – **Gaussian**: $\varphi(\|x_i - c_i\|) = e^{-\sigma\|x_i - c_i\|}$

  – **Logistic**: $\varphi(\|x_i - c_i\|) = \frac{1}{1+e^{-\sigma\|x_i - c_i\|}}$

  – **Quadratic**: $\varphi(\|x_i - c_i\|) = \sigma\|x_i - c_i\|^2$

  Η νόρμα είναι η Ευκλείδια νόρμα και η σ είναι παράμετρος.

- **Παράμετρος σ.** Η παράμετρος σ είναι ίση με:

  $$\sigma = \frac{d}{\sqrt{2k}}$$

  όπου d είναι η μέγιστη απόσταση ανάμεσα σε δύο κέντρα και k το πλήθος των κέντρων.

- **Παράμετρος k.** Καθορίζει το πλήθος των νευρώνων του κρυφού στρώματος και συνεπώς και το πλήθος των κέντρων.

- **Τρόπος επιλογής κέντρων**. Δίνεται δυνατότητα επιλογής των κέντρων με δύο τρόπους:

  i) Χρήση του αλγόριθμου **K-means**. Λαμβάνεται έτοιμος από τη βιβλιοθήκη **scikit-learn**.

  ii) Επιλογή k - το πλήθος κέντρων από το data set τυχαία.

- **Μείωση διάστασης των δεδομένων εισόδου.** Μειώνεται η διάσταση των δεδομένων εισόδου με εφαρμογή **PCA** (Principal Component Analysis) στη διάσταση που καθορίζεται από την παράμετρο *n_components*.

Στο τέλος της λειτουργίας του προγράμματος και αφότου έχει βρεθεί η απόδοση στο train και στο test, τυπώνονται 3 χαρακτηριστικά παραδείγματα ορθής και εσφαλμένης κατηγοριοποίησης αντίστοιχα.

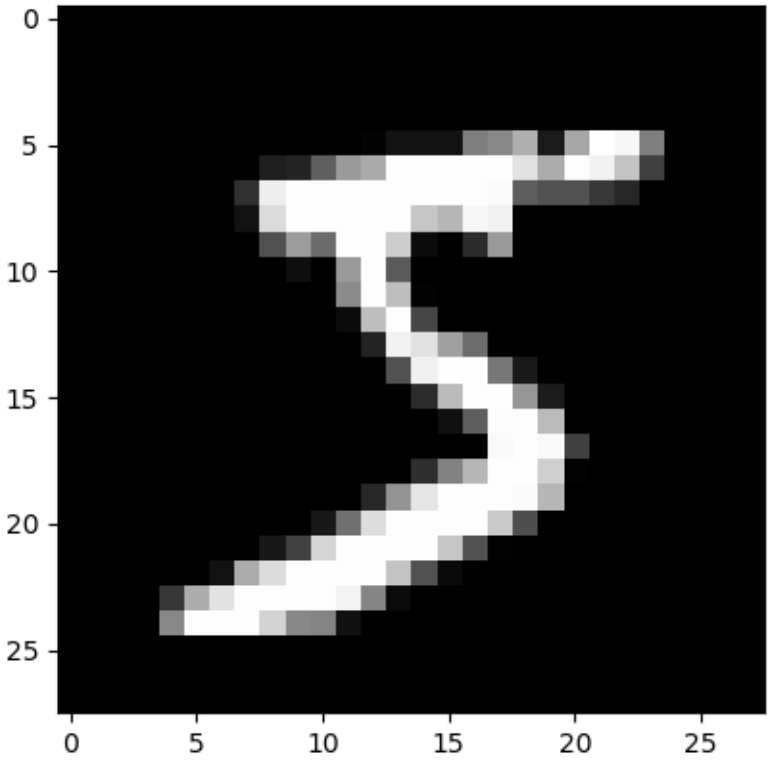
Πειράματα

Έγινε πειραματισμός με όλες τις παραμέτρους του προγράμματος. Τα μεγαλύτερα ποσοστά επιτυχημένης κατηγοριοποίησης επιτεύχθηκαν με χρήση της **Logistic Function** ως συνάρτηση ενεργοποίησης του κρυφού επιπέδου, επιλογή $k = 10000$ κέντρων μέσω του αλγορίθμου **K-means** και μειωμένη διάσταση εισόδου στις 50.
Με τις παραμέτρους αυτές η εκπαίδευση διήρκησε σχεδόν δυόμιση ώρες, όπως φαίνεται στο screenshot παρακάτω, και επιτεύχθηκαν ποσοστά:

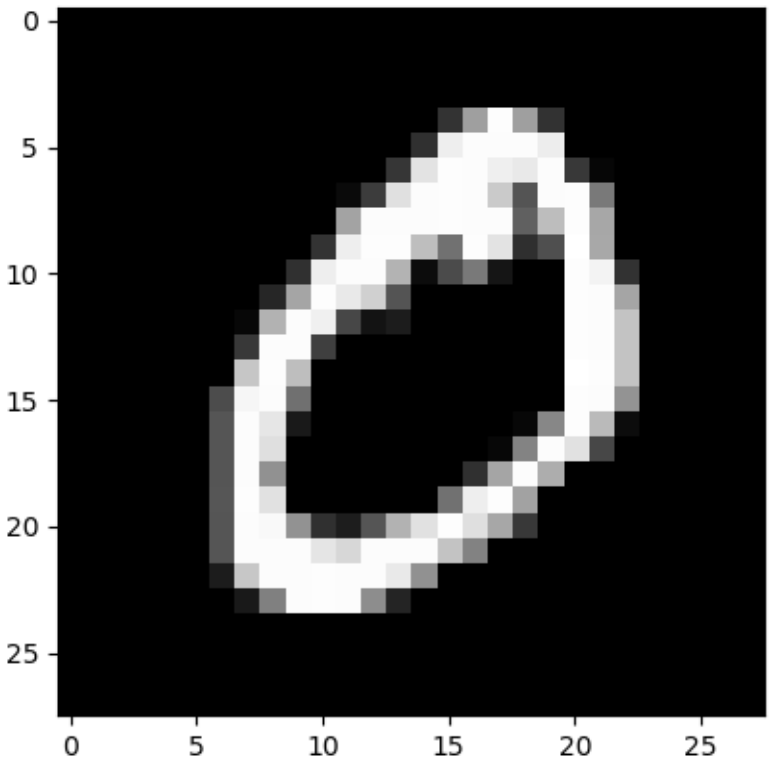
Train: 99.4%, **Test: 98.5%**

```
Data dimensionality: (50000, 50)
Hidden layer size: 10000
Finding centers using K-means algorithm
Radial function: Logistic
sigma = 0.10922702407280342
Elapsed time for training: 8435.544944286346
Accuracy on train: 0.99414
Accuracy on test: 0.9851
```

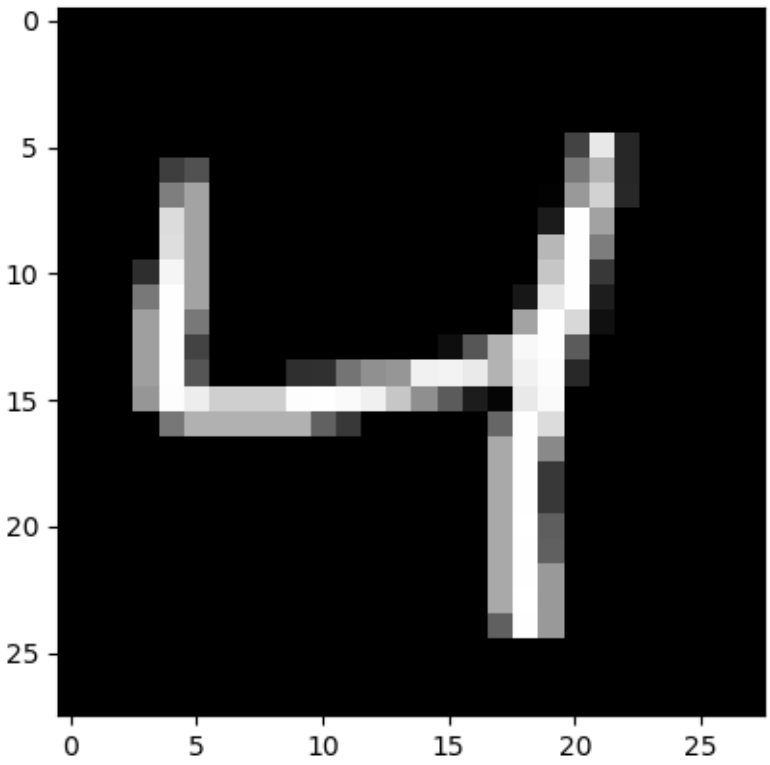
Παρακάτω τρία χαρακτηριστικά παραδείγματα ορθής κατηγοριοποίησης:



(1)

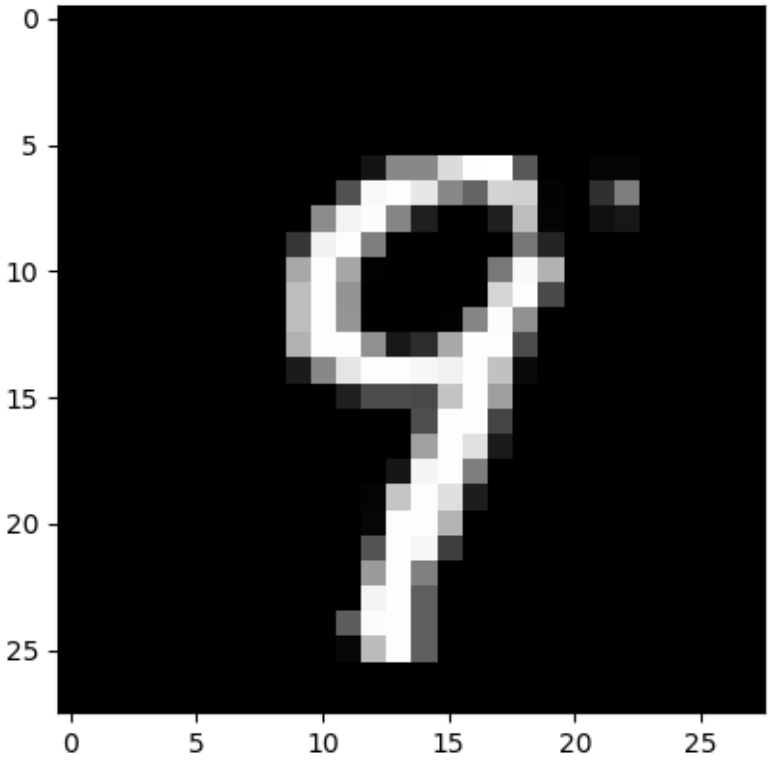


(2)

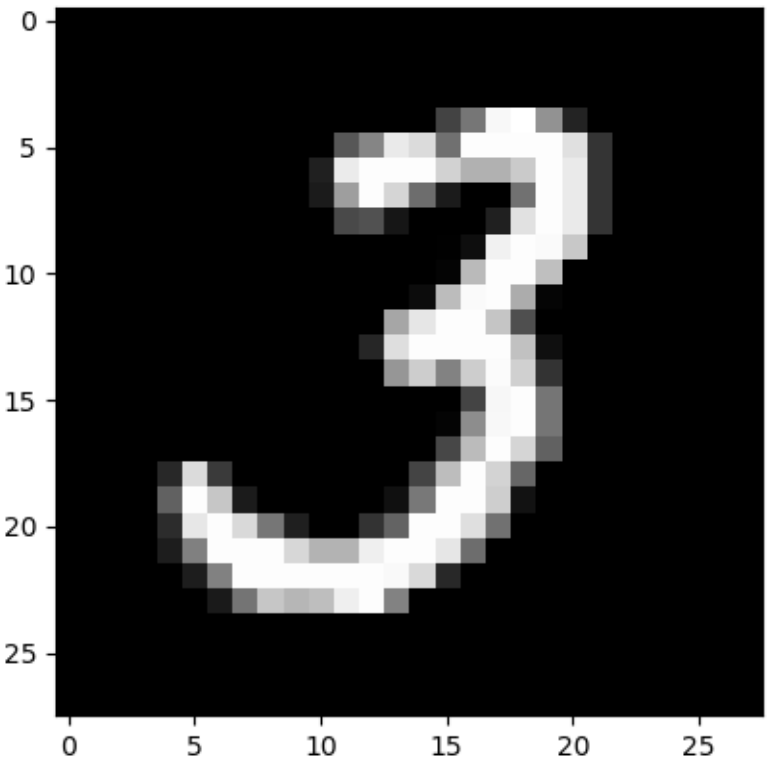


(3)

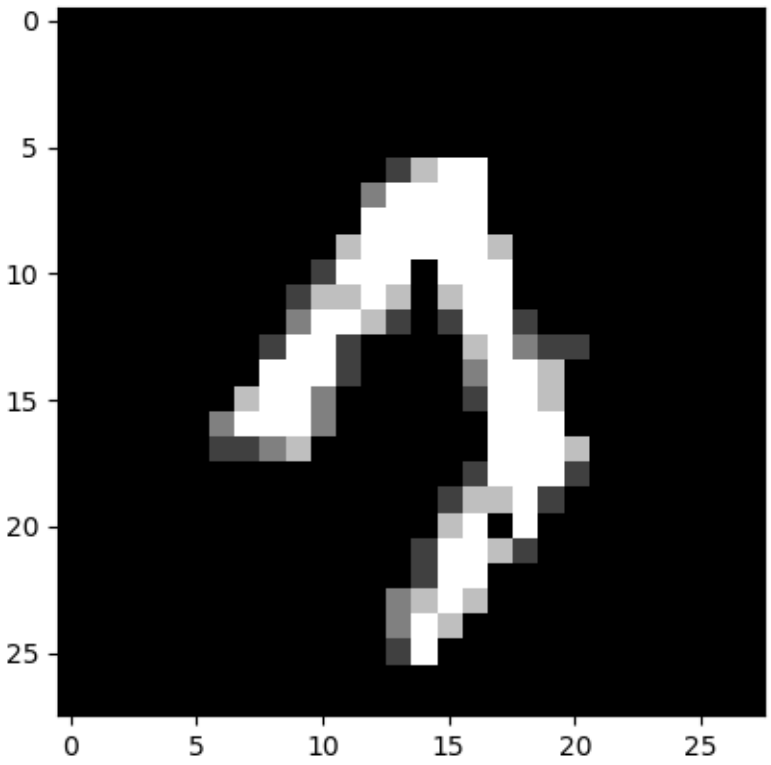
Στη συνέχεια τρία χαρακτηριστικά παραδείγματα εσφαλμένης κατηγοριοποίησης:



(1)



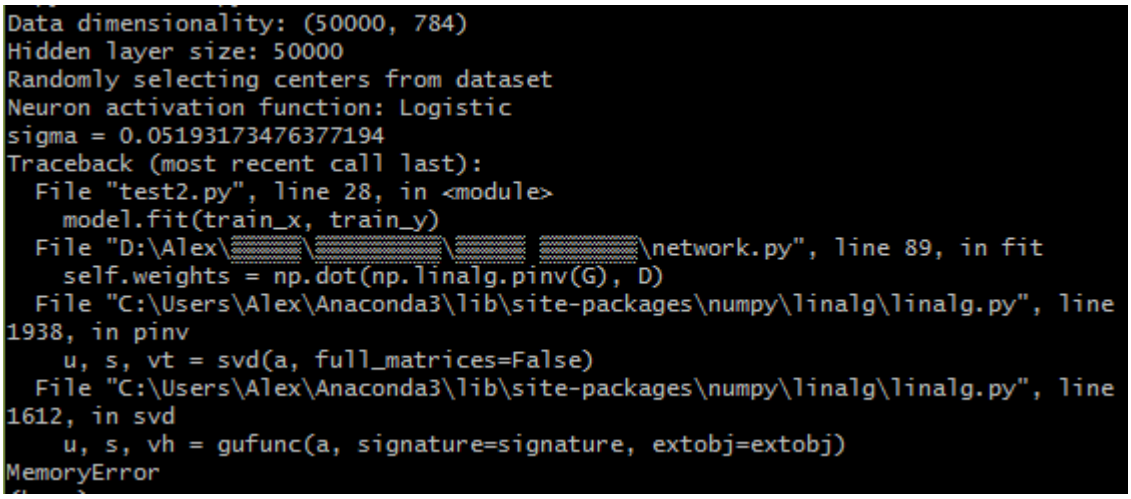
(2)



(3)

Ειδικά στο τελευταίο παράδειγμα βλέπουμε ότι δεν ευθύνεται το πρόγραμμα για το σφάλμα, καθώς ούτε ένας άνθρωπος θα μπορούσε να αναγνωρίσει αυτό το ψηφίο.

Δεν κατάφερα να τρέξω το πρόγραμμα με $k = 50000$, δηλαδή το κάθε data point του train data set να είναι και κέντρο, καθώς μου έβγαλε σφάλμα "**MemoryError**".



Ακολουθούν τα υπόλοιπα πειράματα που έκανα, κατηγοριοποιημένα με βάση τη συνάρτηση ενεργοποίησης που χρησιμοποίησα κάθε φορά:

- **Quadratic Function**

$d = 50$, $k = 100$, center selection algorithm: **random**

```
Data dimensionality: (50000, 50)
Hidden layer size: 100
Randomly selecting centers from dataset
Neuron activation function: Quadratic
sigma = 1.0347350720636306
Elapsed time for training: 36.015761375427246
Accuracy on train: 0.85442
Accuracy on test: 0.8675
```

$d = 50$, $k = 1000$, center selection algorithm: **random**

```
Data dimensionality: (50000, 50)
Hidden layer size: 1000
Randomly selecting centers from dataset
Neuron activation function: Quadratic
sigma = 0.3205547823212483
Elapsed time for training: 376.8678677082062
Accuracy on train: 0.86328
Accuracy on test: 0.8636
```

$d = 784$, $k = 1000$, center selection algorithm: **K-means**

```
Data dimensionality: (50000, 784)
Hidden layer size: 1000
Finding centers using K-means algorithm
Radial function: Quadratic
sigma = 0.288728319710667
Elapsed time for training: 3520.274442911148
Accuracy on train: 0.87346
Accuracy on test: 0.8716
```

$d = 784$, $k = 5000$, center selection algorithm: **random**

```
Data dimensionality: (50000, 784)
Hidden layer size: 5000
Randomly selecting centers from dataset
Radial function: Quadratic
sigma = 0.1579951286315918
Elapsed time for training: 2361.3430709838867
Accuracy on train: 0.90702
Accuracy on test: 0.8604
```

$d = 784$, $k = 10000$, center selection algorithm: **random**

```
Data dimensionality: (50000, 784)
Hidden layer size: 10000
Randomly selecting centers from dataset
Radial function: Quadratic
sigma = 0.1133669572733739
Elapsed time for training: 5439.834805011749
Accuracy on train: 0.9445
Accuracy on test: 0.8426
```

Βλέπουμε ότι το μόνο που καταφέρνουμε με την προσθήκη επιπλέον κέντρων είναι να κάνουμε overfit στο train data set, καθώς τα ποσοστά στο test παραμένουν σταθερά γύρω στο 86%.

- **Gaussian Function**

$d = 50$, $k = 100$, center selection algorithm: **K-means**

```
Data dimensionality: (50000, 50)
Hidden layer size: 100
Finding centers using K-means algorithm
Radial function: Gaussian
sigma = 0.7984471615395266
Elapsed time for training: 107.53945183753967
Accuracy on train: 0.8702
Accuracy on test: 0.8794
```

$d = 50$, $k = 1000$, center selection algorithm: **K-means**

```
Data dimensionality: (50000, 50)
Hidden layer size: 1000
Finding centers using K-means algorithm
Radial function: Gaussian
sigma = 0.2875419994281477
Elapsed time for training: 960.5982608795166
Accuracy on train: 0.95394
Accuracy on test: 0.9571
```

$d = 784$, $k = 5000$, center selection algorithm: **random**

```
Data dimensionality: (50000, 784)
Hidden layer size: 5000
Randomly selecting centers from dataset
Radial function: Gaussian
sigma = 0.1581169033050537
Elapsed time for training: 2793.7124876976013
Accuracy on train: 0.94572
Accuracy on test: 0.9403
```

Εδώ τα έχουμε πάει καλύτερα, με **train** 95.4% και **test** 95.7%.

- **Logistic Function**

Πρώτα μία προσπάθεια με μόλις 10 κέντρα:
$d = 50$, $k = 10$, center selection algorithm: **K-means**

```
Data dimensionality: (50000, 50)
Hidden layer size: 10
Finding centers using K-means algorithm
Neuron activation function: Logistic
sigma = 1.7649304020218677
Elapsed time for training: 12.280730962753296
Accuracy on train: 0.55492
Accuracy on test: 0.5564
```

Έχουμε μόλις 55% ποσοστό επιτυχίας, κάτι που ήταν αναμενόμενο, καθώς τα RBF νευρωνικά δίκτυα πρέπει με βάση τη θεωρία να αυξήσουν τη διάσταση της εισόδου και όχι να τη μειώσουν.

Μία προσπάθεια με ίσο αριθμό κέντρων και διάσταση εισόδου:
$d = 50$, $k = 50$, center selection algorithm: **K-means**

```
Data dimensionality: (50000, 50)
Hidden layer size: 50
Finding centers using K-means algorithm
Neuron activation function: Logistic
sigma = 1.0191954612731933
Elapsed time for training: 50.39983034133911
Accuracy on train: 0.83938
Accuracy on test: 0.8491
```

Τα έχουμε πάει καλά ακόμα και χωρίς αύξηση της διάστασης εισόδου, αλλά δεν είμαστε κοντά στο 98.5% που πιάνουμε με τις καλύτερες δυνατές παραμέτρους.

Αυξάνοντας τη διάσταση στο διπλάσιο έχουμε κατευθείαν πολύ καλύτερο αποτέλεσμα μεγαλύτερο του 90%:
$d = 50$, $k = 100$, center selection algorithm: **K-means**

```
Data dimensionality: (50000, 50)
Hidden layer size: 100
Finding centers using K-means algorithm
Neuron activation function: Logistic
sigma = 0.803945268503991
Elapsed time for training: 91.34048199653625
Accuracy on train: 0.90412
Accuracy on test: 0.9116
```

Αυξάνουμε κι άλλο το πλήθος των κέντρων:

$d = 784$, $k = 1000$, center selection algorithm: **K-means**

```
Data dimensionality: (50000, 784)
Hidden layer size: 1000
Finding centers using K-means algorithm
Neuron activation function: Logistic
sigma = 0.28650817283332586
Elapsed time for training: 4134.302861690521
Accuracy on train: 0.96762
Accuracy on test: 0.9654
```

$d = 784$, $k = 5000$, center selection algorithm: **K-means**

```
Data dimensionality: (50000, 784)
Hidden layer size: 5000
Finding centers using K-means algorithm
Neuron activation function: Logistic
sigma = 0.15546188354492188
Elapsed time for training: 18707.201696395874
Accuracy on train: 0.98696
Accuracy on test: 0.9802
```

Στο τελευταίο παράδειγμα φτάνουμε πολύ κοντά στο μέγιστο ποσοστό, και βλέπουμε οτί ο χρόνος εκπαίδευσης έχει αυξηθεί σημαντικά, περίπου στις 5 ώρες.

Αποτελέσματα

Το πρόγραμμα τα πήγε πολύ καλά στην αναγνώριση των χειρόγραφων ψηφίων. Συγκεκριμένα τα μεγαλύτερα ποσοστά επιτυχίας με τις τρεις συναρτήσεις ενεργοποίησης ήταν:

Quadratic: Train 87.3%**, Test** 87.2%**.**
Gaussian: Train 95.4%**, Test** 95.7%**.**
Logistic: Train 99.4%**, Test** 98.5%**.**

Θεωρώντας την απόδοση του RBF δικτύου ως αυτή που πέτυχα με την **Logistic Function**, συγκρίνουμε με τα αποτελέσματα της ενδιάμεσης εργασίας. Εκεί είχαμε:

KNeighbors classifier: 97% (για 3 γείτονες)
NearestCentroid classifier: 82%

Συμπεραίνουμε ότι το RBF δίκτυο τα έχει πάει καλύτερα και από τους δύο classifiers.