Νευρωνικά Δίκτυα - Βαθιά Μάθηση
Τρίτη Εργασία

4 Απριλίου 2024

Δημήτριος Αλεξόπουλος
aadimitri@ece.auth.gr
ΑΕΜ 10091

Περιεχόμενα

Κατάλογος Σχημάτων

1 Εισαγωγή

Σκοπός της παρούσας εργασίας είναι η υλοποίηση ενός Δικτύου Συνάρτησης Βάσης Ακτινικού Τύπου (Radial Basis Function Neural Network) που θα εκπαιδευτεί σε κάποιο dataset για να λύσει το πρόβλημα της κατηγοριοποίησης 10 κλάσεων.
Για τους σκοπούς της εργασίας επιλέγεται η βάση δεδομένων $CIFAR-10$, η οποία αποτελείται από 60000 έγχρωμες εικόνες 32×32 σε 10 κλάσεις, με 6000 εικόνες ανά κλάση. Υπάρχουν 50000 εικόνες εκπαίδευσης και 10000 εικόνες δοκιμής.
Το σύνολο δεδομένων χωρίζεται σε πέντε παρτίδες εκπαίδευσης και μία παρτίδα δοκιμής, κάθε μία με 10000 εικόνες. Η παρτίδα δοκιμής περιέχει ακριβώς 1000 τυχαία επιλεγμένες εικόνες από κάθε κλάση. Οι παρτίδες εκπαίδευσης περιέχουν τις υπόλοιπες εικόνες με τυχαία σειρά, αλλά ορισμένες παρτίδες εκπαίδευσης μπορεί να περιέχουν περισσότερες εικόνες από μια κλάση από ό,τι από μια άλλη. Μεταξύ τους, οι παρτίδες εκπαίδευσης περιέχουν ακριβώς 5000 εικόνες από κάθε κλάση.
Παρακάτω θα υλοποιήσουμε from scratch σε python το Radial Basis Function Neural Network, παρουσιάζοντας τόσο το θεωρητικό υπόβαθρο, όσο και τα αποτελέσματα της υλοποίησης.

2 Υλοποίηση

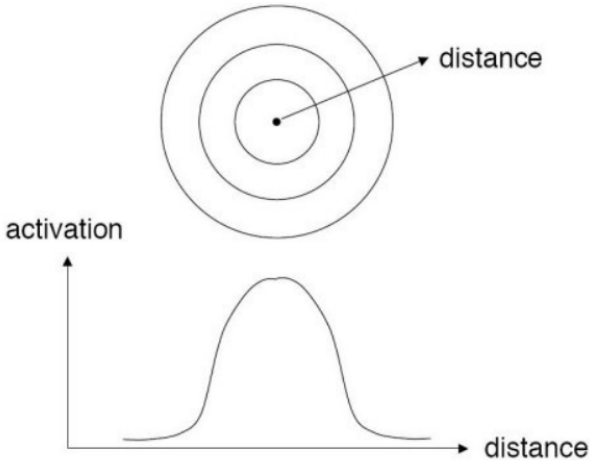
Θα περιγράψουμε, αρχικά, το μαθηματικό υπόβαθρο της υλοποίησης ενός Radial Basis Function Neural Network. Παράλληλα με την θεωρητική ερμηνεία θα υπάρχει κι επεξήγηση του υλοποιημένου κώδικα σε python. Σημειώνουμε ότι για την μείωση της πολυπλοκότητας χρησιμοποιήθηκε ο αλγόριθμος PCA (Principal Component Analysis) και εντοπίστηκε ο αριθμός των features που διατηρούν το 90% της πληροφορίας ως $num\_components = 103$.

2.1 Συνάρτηση Ακτινικού Τύπου

Αν και η υλοποίηση είναι διαφορετική, τα νευρωνικά δίκτυα RBF είναι εννοιολογικά παρόμοια με τα μοντέλα K-Nearest Neighbor (k-NN). Η βασική ιδέα είναι κι εδώ ότι ένα νέο στοιχείο είναι πιθανό να ανήκει στην ίδια κλάση με άλλα στοιχεία με τα οποία έχει κοντινές τιμές στα χαρακτηριστικά του (features). Για την αυτοματοποίηση αυτής της ιδέας υπολογίζεται η ευκλείδεια απόσταση από το σημείο που αξιολογείται έως το κέντρο κάθε νευρώνα και μια συνάρτηση βάσης ακτινικού τύπου (RBF) (που ονομάζεται επίσης συνάρτηση πυρήνα) εφαρμόζεται στην απόσταση για τον υπολογισμό του βάρους - επιρροής (weight) για κάθε νευρώνα. Η συνάρτηση βάσης ακτινικού τύπου ονομάζεται έτσι επειδή η ακτινική απόσταση (distance) είναι το όρισμα της συνάρτησης:

$$Weight = RBF(distance)$$

Όσο πιο μακριά βρίσκεται ένας νευρώνας από το νέο στοιχείο ενδιαφέροντός μας (το νέο στοιχείο που θέλουμε να κατηγοριοποιήσουμε), τόσο μικρότερη επιρροή έχει σαυτό:



Σχήμα 1: Συνάρτηση βάσης ακτινικού τύπου

Όπως βλέπουμε και στο παραπάνω σχήμα, η πιο συνηθισμένη συνάρτηση βάσης ακτινικού τύπου είναι κάποια συνάρτηση γκαουσιανής μορφής. Στην υλοποιημένη συνάρτησή μας gaussian_rbf(x, center) χρησιμοποιείται η μορφή:

$$RBF(x, center) = e^{-\gamma \|x - center\|^2}$$

όπου η υπερπαράμετρος γ καθορίζει το εύρος της γκαουσιανής καμπύλης και θα πρέπει να ρυθμιστεί κατάλληλα.

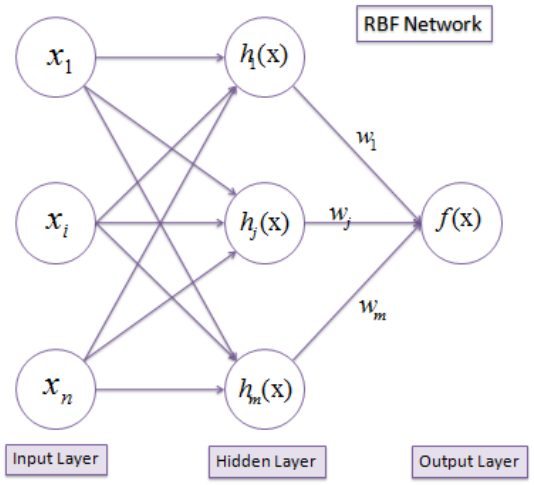
2.2 Τοπολογία Δικτύου *RBF*

Τα δίκτυα *RBF*, σε αντίθεση με τα δίκτυα πολυστρωματικού *perceptron* που μελετήσαμε στην πρώτη εργασία, απαρτίζονται αυστηρά από τρία στρώματα:

- Στρώμα εισόδου - Υπάρχει ένας νευρώνας για κάθε *feature*. Στη συνέχεια, οι νευρώνες εισόδου τροφοδοτούν τις τιμές σε καθέναν από τους νευρώνες στο κρυφό στρώμα.

- Κρυφό στρώμα - Αυτό το στρώμα έχει μεταβλητό αριθμό νευρώνων (ο βέλτιστος αριθμός καθορίζεται από τη διαδικασία εκπαίδευσης). Κάθε νευρώνας αποτελείται από μια συνάρτηση βάσης ακτινικού τύπου με κέντρο ένα σημείο με τόσες διαστάσεις όσες είναι οι μεταβλητές πρόβλεψης (features). Η εξάπλωση (ακτίνα) της συνάρτησης *RBF* μπορεί να είναι διαφορετική για κάθε διάσταση. Τα κέντρα και τα *spreads* καθορίζονται από τη διαδικασία εκπαίδευσης. Η προκύπτουσα τιμή διαβιβάζεται στο στρώμα άθροισης.

- Στρώμα άθροισης - Η τιμή που βγαίνει από έναν νευρώνα στο κρυφό στρώμα πολλαπλασιάζεται με ένα βάρος που σχετίζεται με τον νευρώνα και περνάει στο στρώμα άθροισης το οποίο αθροίζει τις σταθμισμένες τιμές και παρουσιάζει αυτό το άθροισμα ως έξοδο του δικτύου. Για προβλήματα ταξινόμησης, υπάρχει μία έξοδος για κάθε κατηγορία-κλάση. Η τιμή εξόδου για μια κλάση είναι η πιθανότητα το νέο στοιχείο που αξιολογείται να ανήκει στην εν λόγω κλάση.



$$f(\mathrm{x}) = \sum_{j=1}^{m} w_j h_j(\mathrm{x})$$

$$h(x) = \exp\left(-\frac{(x-c)^2}{r^2}\right)$$

Σχήμα 2: Δομή δικτύου *RBF*

Η έξοδος του κρυφού στρώματος που αναλύθηκε παραπάνω υπολογίζεται από την συνάρτησή μας calculate_hidden_outputs(x), όπου x είναι ένα σύνολο δεδομένων εισόδου.

2.3 Εκπαίδευση

Οι ακόλουθες παράμετροι καθορίζονται από τη διαδικασία εκπαίδευσης:

- Ο αριθμός των νευρώνων στο κρυφό στρώμα.
- Οι συντεταγμένες του κέντρου κάθε συνάρτησης *RBF* του κρυφού στρώματος.
- Η ακτίνα (εξάπλωση) κάθε συνάρτησης *RBF* σε κάθε διάσταση.
- Τα βάρη που εφαρμόζονται στις εξόδους της συνάρτησης *RBF* καθώς περνούν στο στρώμα άθροισης.

Είναι σημαντικό να σημειώσουμε ότι το κρυφό στρώμα εκπαιδεύεται ξεχωριστά από το εξωτερικό στρώμα (άθροισης).

2.3.1 Κρυφού Στρώματος (Κέντρα)

Μία προσέγγιση είναι απλώς η τυχαία επιλογή κέντρων, όπως αυτή υλοποιείται από την συνάρτηση random_centers().
Μία άλλη πιο ενδελεχής προσέγγιση είναι η ομαδοποίηση $K-means$ για την εύρεση των κέντρων των συστάδων, τα οποία στη συνέχεια χρησιμοποιούνται ως κέντρα για τις συναρτήσεις *RBF*. Ωστόσο, η ομαδοποίηση $K-means$ είναι μια διαδικασία με μεγάλη υπολογιστική πολυπλοκότητα και συχνά δεν παράγει τον βέλτιστο αριθμό κέντρων. Η λειτουργία της υλοποιείται στην συνάρτησή μας kmeans_centers() και περιγράφεται από τον παρακάτω ψευδοκώδικα:

```
Αρχικοποίησε τα κέντρα σε τυχαίες τιμές.
Επανάλαβε «
```

```
    Για κάθε πρότυπο «
        Βρες το κοντινότερο κέντρο σε αυτό
    »
    Για κάθε κλάση  «
        Θέσε το νέο κέντρο της ως τον μέσο όρο των προτύπων τα οποία
        ανήκουν στην κλάση
    »
» Μέχρι να μην υπάρξει καμία αλλαγή στα κέντρα.
```

2.3.2 Εξωτερικού Στρώματος (Βάρη)

Εδώ χρησιμοποιείται ο κλασικός κανόνας Δέλτα που αναλύθηκε και στην πρώτη εργασία. Με το forward propagation υπολογίζουμε την τιμή της εξόδου του δικτύου, καθώς η έξοδος κάθε στρώματος αποτελεί την είσοδο του επόμενου στρώματος. Η τιμή αυτή προκύπτει με είσοδο το διάνυσμα των χαρακτηριστικών (features) του κάθε δεδομένου. Στη συνέχεια το αποτέλεσμα περνά μέσα από μια συνάρτηση ενεργοποίησης (εδω softmax), ώστε να αποφασιστεί τελικά η κλάση. Αυτό συμβαίνει διότι η τιμή εξόδου για μια κλάση είναι μόνο η πιθανότητα το νέο στοιχείο που αξιολογείται να ανήκει στην εν λόγω κλάση κι επομένως έχει μη μηδενική τιμή για κάθε κλάση. Η συνάρτηση ενεργοποίησης αναδεικνύει ποιός νευρώνας του στρώματος εξόδου (κλάση) τελικά θα ενεργοποιηθεί:

```
output = np.dot(hidden_outputs, self.weights)
output = self.softmax(output)
```

Με βάση το αποτέλεσμα της εξόδου υπολογίζεται το σφάλμα (error) σε σχέση με την γνωστή ετικέτα του συνόλου εκπαίδευσης για το εκάστοτε δεδομένο. Έτσι, μπορούμε απλά με μια μέθοδο gradient descent να εντοπίσουμε τα βάρη εκείνα που ελαχιστοποιούν το σφάλμα:

```
error = output - y_train
gradient = np.dot(hidden_outputs.T, error)
self.weights -= learning_rate * gradient
```

Η υπερπαράμετρος learning_rate είναι ο ρυθμός εκμάθησης και έπεται να προσδιοριστεί. Οι δύο διαδικασίες εκπαίδευσης συνοψίζονται στην υλοποιημένη συνάρτηση fit().

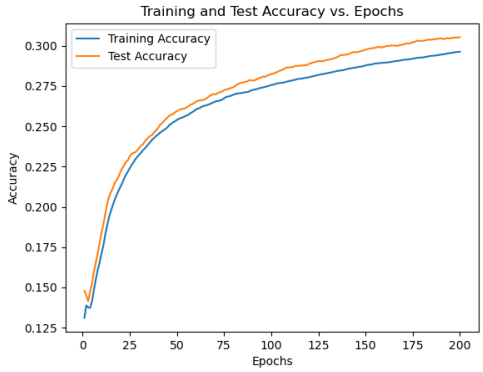
3 Αποτελέσματα - Σχολιασμοί

Ακολουθούν τα αποτελέσματα της παραπάνω θεωρητικής ανάλυσης και της υλοποίησης του κώδικα σε *python*. Θα δοθούν χαρακτηριστικά παραδείγματα ορθής και εσφαλμένης κατηγοριοποίησης, καθώς και ποσοστά επιτυχίας στα στάδια της εκπαίδευσης (*training*) και του ελέγχου (*testing*), χρόνος εκπαίδευσης και ποσοστά επιτυχίας για διαφορετικούς αριθμούς νευρώνων στο κρυφό επίπεδο, διαφορετικό τρόπο εκπαίδευσης (K-μέσους, τυχαία επιλογή κέντρων), διαφορετικές τιμές των παραμέτρων εκπαίδευσης. Επιπλέον, θα συγκριθεί η απόδοση του νευρωνικού σε σχέση με την κατηγοριοποίηση πλησιέστερου γείτονα (*Nearest Neighbor*) και πλησιέστερου κέντρου κλάσης (*Nearest Class Centroid*) της ενδιάμεσης εργασίας. Η ορθή και εσφαλμένη κατηγοριοποίηση μπορεί να δειχθεί με την μορφή:

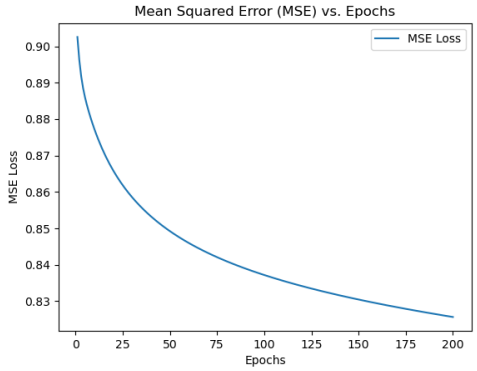
```
pred: 5  true: 5
pred: 2  true: 8
pred: 4  true: 4
pred: 8  true: 0
pred: 6  true: 3
pred: 1  true: 1
```

3.1 Δοκιμή Αριθμών Νευρώνων στο Κρυφό Στρώμα

Εκπαιδεύουμε το νευρωνικό δίκτυο για αριθμό νευρώνων στο κρυφό στρώμα $n = [20, 50, 80]$. Θέτουμε $learning\_rate = 0.0001$, $learning\_alg = kmeans$ και εκπαιδεύουμε το δίκτυο για 200 εποχές. Παρακάτω φαίνονται ενδεικτικά τα διαγράμματα του MSE και $accuracy$ στην περίπτωση του $n = 80$:



Σχήμα 3: Απόδοση - n



Σχήμα 4: Μέσο Τετραγωνικό Σφάλμα - n

Τα πλήρη (ενδεικτικά) αποτελέσματα φαίνονται συνοπτικά στον παρακάτων πίνακα:

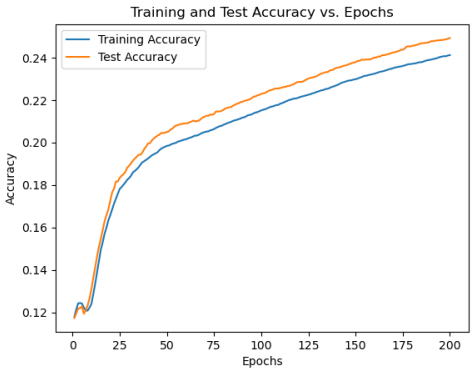
n	$Accuracy$	MSE	$Execution\ Time$
20	0.26	0.85	22.08s
50	0.29	0.83	49.75s
80	0.31	0.83	80.73s

Πίνακας 1: Ενδεικτική απόδοση και σφάλμα για διαφορετικό αριθμό νευρώνων του κρυφού στρώματος

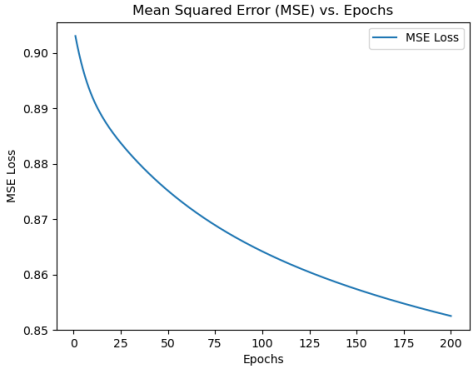
Παρατηρήσεις:
Παρατηρούμε ότι όσο αυξάνουμε τον αριθμό νευρώνων του κρυφού στρώματος αυξάνεται η απόδοση του δικτύου. Γενικά δεν είναι κανόνας οπότε ακολουθείται τεχνική trial and error. Το βέβαιο είναι ότι αυξάνεται ο χρόνος εκτέλεσης, καθώς αυξάνεται η πολυπλοκότητα του δικτύου (περισσότεροι νευρώνες, άρα περισσότερες συνδέσεις, επομένως περισσότερα $weights$ και μεγαλύτερο υπολογιστικό κόστος).

3.2 Δοκιμή Τρόπου Εκπαίδευσης Κρυφού Στρώματος

Εκπαιδεύουμε το νευρωνικό δίκτυο με τυχαία επιλογή κέντρων έναντι του αλγόριθμου K-μέσων που χρησιμοποιήθηκε προηγουμένως. Παρακάτω φαίνονται ενδεικτικά τα διαγράμματα του *MSE* και *accuracy* στην περίπτωση του $n = 80$:



Σχήμα 5: Απόδοση - *random*



Σχήμα 6: Μέσο Τετραγωνικό Σφάλμα - *random*

Συγκριτικά έχουμε δηλαδή:

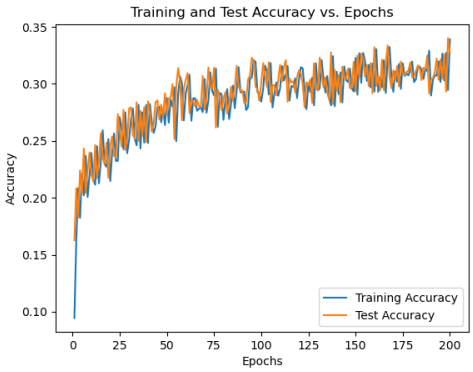
Algorithm	*Accuracy*	*MSE*	*Execution Time*
$k-means$	0.31	0.83	80.73*s*
random	0.25	0.85	58.55*s*

Πίνακας 2: Ενδεικτική απόδοση και σφάλμα για διαφορετικό αλγόριθμο εύρεσης κέντρων

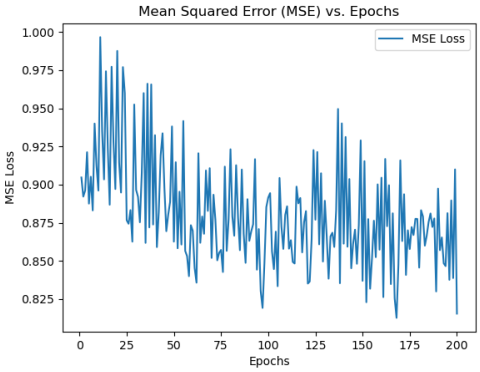
Παρατηρήσεις:
Παρατηρούμε ότι η τυχαία εύρεση κέντρων παρουσιάζει αρκετά μικρότερη επίδοση, όπως ήταν αναμενόμενο. Παρ΄ όλ΄ αυτά, η υπολογιστική πολυπλοκότητα, κι άρα ο χρόνος εκτέλεσης, μειώνεται. Λογικό, εφόσον ο αλγόριθμος K-μέσων είναι απαιτητικός υπολογιστικά.

3.3 Δοκιμή Διάφορων *learning rates*

Εκπαιδεύουμε το νευρωνικό δίκτυο με $learning\_alg = kmeans$ και $n = 80$. Δοκιμάζουμε $learning\_rate = [0.01, 0.001, 0.0001]$ και εκπαιδεύουμε το δίκτυο για 200 εποχές. Παρακάτω φαίνονται ενδεικτικά τα διαγράμματα του *MSE* και *accuracy* στην περίπτωση του $learning\_rate = 0.001$:



Σχήμα 7: Απόδοση - *learning_rate*



Σχήμα 8: Μέσο Τετραγωνικό Σφάλμα - *learning_rate*

Τα πλήρη (ενδεικτικά) αποτελέσματα φαίνονται συνοπτικά στον παρακάτων πίνακα:

learning_rate	*Accuracy*	*MSE*	*Execution Time*
0.01	0.34	0.82	64.32*s*
0.001	0.33	0.82	65.95*s*
0.0001	0.31	0.83	80.73*s*

Πίνακας 3: Ενδεικτική απόδοση και σφάλμα για διαφορετικό *learning_rate*

Παρατηρήσεις:
Παρατηρούμε ότι μεγαλύτερο *learning rate* μπορεί να δώσει καλύτερη επίδοση, ωστόσο προσδίδει μεγαλύτερες διακυμάνσεις μεταξύ της κάθε εποχής. Επομένως, ο αλγόριθμος γίνεται λιγότερο εύρωστος.

3.4 Σύγκριση με Άλλους Κατηγοριοποιητές

Μπορούμε, τέλος, να συγκρίνουμε την απόδοση του νευρωνικού μας δικτύου με τους κατηγοριοποιητές $K-Nearest-Neighbors$ και *Nearest Centroid Classifier* που υλοποιήθηκαν στην ενδιάμεση εργασία. Χρησιμοποιήθηκε το $RBF-NN$ με παραμέτρους $learning\_alg = kmeans$, $n = 80$, $learning\_rate = 0.0001$ και εκπαιδεύτηκε για 200 εποχές:

Classifier	**Accuracy**	**Execution Time**
KNN	0.3860	68.95*s*
NCC	0.2807	0.14*s*
$RBF-NN$	0.31	74.44*s*

Πίνακας 4: Σύγκριση απόδοσης με άλλους κατηγοριοποιητές

Παρατηρήσεις:
Παρατηρούμε ότι με το RBF νευρωνικό δίκτυό μας δεν πετυχαίνουμε καλύτερη απόδοση από το KNN, ωστόσο πετυχαίνομαι καλύτερη από το NCC. Ωστόσο, είναι υπολογιστικά αποδοτικό, καθώς ο χρόνος εκτέλεσης είναι κοντά στον KNN.

Αναφορές

[1] CIFAR-10 and CIFAR-100 datasets - *Canadian Institute for Advanced Research (CIFAR)*. https://www.cs.toronto.edu/~kriz/cifar.html

[2] K-nearest neighbors algorithm - *Wikipedia*. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[3] Nearest centroid classifier - *Wikipedia*. https://en.wikipedia.org/wiki/Nearest_centroid_classifier

[4] Radial Basis Function Neural Networks - *DTREG*. https://www.dtreg.com/solution/rbf-neural-networks

[5] Radial Basis Function Artificial Neural Networks - *YouTube*. https://www.youtube.com/watch?v=OUtTI99uRf4

[6] Diamantaras Slides