**Υπολογιστική Νοημοσύνη Αναφορά 1ης Εργασίας**

Σεργιάννης Παρασκευάς - Βασίλειος  
ΑΜ: 1067467  
Έτος: 5ο

*Github Link: https://github.com/PSergiannis/Computational\_Intelligence2023.git*

*Σημείωση:*  
Για την παραγωγή όλων των αρχείων πρέπει να τρέξετε με τη σειρά το **preprocessing.py** και στη συνέχειατα **centering.py**, **normalization.py** και **standardization.py**, και τα αρχεία να είναι όλα στον ίδιο φάκελο όπως φαίνεται στο παρακάτω screenshot. Επίσης στα αρχεία όπως το **Neural\_Network\_L2.py** αλλάζετε τις τιμές των l2 χειροκίνητα αν θέλετε να τεστάρετε τα αποτελέσματα, σε κάποια έχω αφήσει το τελευταίο παράδειγμα  
(πχ model.add(Dense(11, input\_dim=17, activation='relu', kernel\_regularizer=l2(0.9)))  
model.add(Dense(5, activation='softmax', kernel\_regularizer=l2(0.9)), εδώ έχω αφήσει το l2=0.9). Για ότι τυχόν απορία έχετε ή αν κάτι δεν το βρίσκετε στο link παρακαλώ επικοινωνήστε μαζί μου στο email.

Graphical user interface, application

Description automatically generated

***Α1. Προεπεξεργασία και Προετοιμασία δεδομένων*(preprocessing.py  
centering.py  
normalization.py  
standardization.py)**

α) Για το συγκεκριμένο πρόβλημα από τη στιγμή που έχουμε κυρίως συνεχή χαρακτηριστικά με διαφορετικές μονάδες και κλίμακες, θεωρώ πως θα ήταν χρήσιμη κάποια από τις μεθόδους Normalization ή Standardization για να γίνει πιο εύκολο να συγκριθούν τα χαρακτηριστικά και να βοηθήσουν στη βελτίωση των επιδόσεων του μοντέλου. Επίσης επειδή το Centering μπορεί να είναι χρήσιμο για την εξάλειψη τυχόν biases που μπορεί να προκύψουν λόγω μη μηδενικών μέσων και συμβάλλει στη βελτίωση της ερμηνευσιμότητας των συντελεστών του μοντέλου, ιδίως σε μοντέλα παλινδρόμησης. Γι αυτό το λόγο θεωρώ πως και όλες οι μέθοδοι θα ήταν σκόπιμοι για το συγκεκριμενο πρόβλημα, καταλήγοντας στο Standardization που είναι ο συνδυασμός και των δύο μεθόδων. Αυτό στην θεωρία, στην πράξη θα τεστάρουμε όλες τις μεθόδους για να δούμε ποιά από αυτές θα μας δώσει τα πιο ακριβή αποτελέσματα.  
  
β) Διασταυρούμενη Επικύρωση (cross-validation): Για να βεβαιωθούμε ότι κάθε fold είναι ισορροπημένο (balanced) ως προς τον αριθμό των δειγμάτων κάθε κλάσης, χρησιμοποιούμε το StratifiedKFold αντί για το απλό KFold.

***Α2. Επιλογή αρχιτεκτονικής  
(Neural\_Network.py)***

1. *Η σημασία των παρακάτω μετρικών περιγράφεται παρακάτω:*
2. *Cross-Entropy (CE):* Αυτή η μέθοδος βοηθά το μοντέλο να μάθει πώς να προβλέπει το σωστό αποτέλεσμα συγκρίνοντας τις προβλέψεις του με τις πραγματικές απαντήσεις. Το μοντέλο προσπαθεί να γίνει καλύτερο στις προβλέψεις του ελαχιστοποιώντας το cross-entropy loss. Όσο μικρότερο είναι το cross-entropy loss, τόσο καλύτερο είναι το μοντέλο στην πρόβλεψη του σωστού αποτελέσματος. Αυτή η μέθοδος είναι κατάλληλη για το πρόβλημά μας επειδή λειτουργεί καλά με τα multiclass προβλήματα και βοηθά το μοντέλο να μάθει τα σωστά μοτίβα στα δεδομένα αισθητήρων.
3. *Μέσο τετραγωνικό σφάλμα (MSE):* Αυτή η μέθοδος μετρά πόσο κοντά είναι οι προβλέψεις του μοντέλου στις πραγματικές τιμές, εξετάζοντας τις μέσες τετραγωνικές διαφορές μεταξύ τους. Ωστόσο, δεν είναι κατάλληλη για το πρόβλημά μας, επειδή έχει σχεδιαστεί για προβλήματα όπου πρέπει να προβλέψουμε μια συνεχή τιμή (όπως ένας αριθμός) και όχι μια ετικέτα κλάσης (όπως τα είδη της δραστηριότητας).
4. *Accuracy:* Αυτή η μέθοδος υπολογίζει το ποσοστό των περιπτώσεων που το μοντέλο προβλέπει σωστά την ανθρώπινη δραστηριότητα. Είναι ένας απλός τρόπος μέτρησης του πόσο καλό είναι το μοντέλο στο να κάνει σωστές προβλέψεις. Ωστόσο, το accuracy δεν είναι η κατάλληλη loss function για την εκπαίδευση του μοντέλου, επειδή δεν παρέχει ομαλή ανατροφοδότηση για να μάθει το μοντέλο. Είναι πιο κατάλληλη για την αξιολόγηση της απόδοσης του μοντέλου μετά την εκπαίδευσή του.

*β)* Για το δεδομένο πρόβλημα ταξινόμησης πολλαπλών κατηγοριών, όπου πρέπει να προβλέψουμε μία από τις πέντε πιθανές δραστηριότητες (sitting-down, standing-up, standing, walking, και sitting), θα χρειαστούμε 5 νευρώνες στο επίπεδο εξόδου.

Κάθε νευρώνας αντιστοιχεί σε μια από τις κλάσεις δραστηριοτήτων και βγάζει ως αποτέλεσμα μια τιμή πιθανότητας για τη συγκεκριμένη κλάση. Η χρήση μιας συνάρτησης ενεργοποίησης όπως η softmax στο επίπεδο εξόδου εξασφαλίζει ότι το άθροισμα των πιθανοτήτων για όλες τις κλάσεις ισούται με 1. Έτσι το μοντέλο θα προβλέψει την κλάση με την υψηλότερη πιθανότητα ως τελική έξοδο.

*γ*) Για τους κρυφούς κόμβους επιλέγουμε τη συνάρτηση ενεργοποίησης ReLU (Rectified Linear Unit). Η ReLU είναι δημοφιλής επειδή είναι εύκολη στη χρήση και λειτουργεί καλά στην εκπαίδευση των νευρωνικών δικτύων.

Η ReLU έχει τα ακόλουθα πλεονεκτήματα:

1. Μη γραμμικότητα (Non-linearity): Βοηθά το νευρωνικό δίκτυο να μαθαίνει σύνθετα μοτίβα.
2. Simplicity: Είναι εύκολη στον υπολογισμό, καθιστώντας τη διαδικασία εκμάθησης ταχύτερη.
3. Sparse activation: Επιτρέπει σε ορισμένους μόνο νευρώνες να είναι ενεργοί κάθε φορά, γεγονός που βοηθά το νευρωνικό δίκτυο να μαθαίνει καλύτερα.
4. Αποφεύγει τα vanishing gradients: Βοηθά το νευρωνικό δίκτυο να συνεχίσει να μαθαίνει ακόμη και όταν αντιμετωπίζει προκλήσεις όπως οι vanishing gradients.

Συνοψίζοντας, η ReLU είναι μια καλή επιλογή σαν συνάρτηση ενεργοποίησης για τους κρυφούς νευρώνες, επειδή είναι απλή, αποτελεσματική και βοηθά το νευρωνικό δίκτυο να μάθει πολύπλοκα μοτίβα στα δεδομένα που μας παρέχονται.  
  
  
δ) Στο επίπεδο εξόδου η καταλληλότερη συνάρτησης ενεργοποίησης είναι η Softmax.

Η Softmax είναι κατάλληλη για multiclass προβλήματα επειδή μετατρέπει την έξοδο κάθε νευρώνα σε μια κατανομή πιθανότητας στις πιθανές κλάσεις. Αυτό σημαίνει ότι το άθροισμα των πιθανοτήτων για όλες τις κλάσεις θα ισούται με 1, καθιστώντας εύκολη την ερμηνεία των αποτελεσμάτων και την επιλογή της κλάσης με την υψηλότερη πιθανότητα ως τελική πρόβλεψη.

Οι άλλες δυο συναρτήσεις ενεργοποίησης που αναφέρθηκαν δεν είναι τόσο κατάλληλες για το επίπεδο εξόδου σε αυτή την περίπτωση:

* Σιγμοειδής: Είναι καταλληλότερη για προβλήματα δυαδικής ταξινόμησης, όπου έχουμε μόνο δύο πιθανές κλάσεις. Δεν είναι ιδανική για προβλήματα ταξινόμησης πολλαπλών κλάσεων όπως το δικό σας.
* Γραμμική: Δεν λειτουργεί τόσο καλά για προβλήματα ταξινόμησης, καθώς δεν παρέχει πιθανότητες ή ένα σαφές όριο απόφασης για τις κλάσεις.

ε) (i)Παρατηρούμε πως ο καλύτερος αριθμός κόμβων στο κρυφό επίπεδο είναι 11 γιατί σε αυτή την περίπτωση έχουμε το μικρότερο CE loss και το μεγαλυτερο accuracy. H αύξηση του αριθμού των κρυφών κόμβων μπορεί να βοηθήσει το μοντέλο να συλλάβει πιο σύνθετα μοτίβα στα δεδομένα, αλλά μετά από ένα σημείο η απόδοση πέφτει όπως και στο παράδειγμά μας.  
  
(ii) Χρησιμοποιούμε την 'sparse\_categorical\_crossentropy' ως συνάρτηση κόστους. Αυτή είναι κατάλληλη για προβλήματα ταξινόμησης πολλαπλών κλάσεων (multiclass), όπου η target variable αποτελείται από πολλαπλές κλάσεις και κάθε instance ανήκει σε μία κλάση. Αυτή η συνάρτηση κόστους υπολογίζει τη διασταυρούμενη εντροπία μεταξύ των πραγματικών class labels και των προβλεπόμενων πιθανοτήτων, βοηθώντας το μοντέλο να κάνει ακριβείς προβλέψεις. Εάν το πρόβλημα ήταν δυαδική ταξινόμηση, η 'binary\_crossentropy' θα ήταν καταλληλότερη. Για προβλήματα regression, θα μπορούσε να χρησιμοποιηθεί το μέσο τετραγωνικό σφάλμα (MSE) ή το μέσο απόλυτο σφάλμα (MAE).  
Στην δικιά μας περίπτωση η 'sparse\_categorical\_crossentropy' είναι καταλληλότερη.  
  
(iii) Η ταχύτητα σύγκλισης σε όλα τα πειράματα είναι μεγαλύτερη στην αρχή που το μοντέλο μαθαίνει από τα πρώτα instances και αργότερα αφού μαθαίνει πιο αργά η ταχύτητα σύγκλισης μειώνεται.

|  |  |  |  |
| --- | --- | --- | --- |
| Αριθμός νευρώνων στο κρυφό επίπεδο | CE loss | ΜSE | Acc |
| Η1 = Ο  (5 nodes) | 1.49824 | 0.64357 | 0.81807 |
| Η1 = (Ι+Ο)/2  (11 nodes) | 1.28739 | 0.48539 | 0.85088 |
| Η1 = Ι+Ο  (22 nodes) | 3.96041 | 0.73513 | 0.82064 |

Η1 = 5  
Chart

Description automatically generated  
Chart

Description automatically generated  
  
  
Η1 = 11  
Chart

Description automatically generated  
  
Chart

Description automatically generated

Η1 = 22  
Chart

Description automatically generated  
  
Chart

Description automatically generated

στ) Ένα κατάλληλο κριτήριο τερματισμού θα μπορούσε να ήταν να παρακολουθούμε μια καθορισμένη μετρική (το accuracy ή το loss) και να σταματάμε την εκπαίδευση όταν η μετρική αυτλη δεν βελτιώνεται για έναν ορισμένο αριθμό διαδοχικών εποχών (θα μπορούσαμε να πούμε 5). Θα μπορούσαμε να χρησιμοποιήσουμε το κριτήριο του προώρου σταματήματος για να αποφύγουμε το overtraining/overfitting του μοντέλου και να γίνει πιο γρήγορη η εκπαίδευση του μοντέλου.

***Α3. Μεταβολές στον ρυθμό εκπαίδευσης και σταθεράς ορμής***

***(Neural\_Network\_SGD.py)***

Ο Adam δεν παρέχει άμεση παράμετρο για την τροποποίηση της τιμής της ορμής m.   
Γι αυτό για την συνέχεια των πειραμάτων θα χρησιμοποιήσουμε το SGD σαν classifier.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| η | m | CE loss | ΜSE | Acc |
| 0.001 | 0.2 | 0.86278 | 0.49477 | 0.78959 |
| 0.001 | 0.6 | 1.01608 | 0.50782 | 0.81197 |
| 0.05 | 0.6 | 4.71857 | 0.33840 | 0.80981 |
| 0.1 | 0.6 | 3.00950 | 0.58672 | 0.82018 |

Παρατηρούμε πως μεταξύ των τεσσάρων μοντέλων, το μοντέλο 1 (η=0.001, m=0.2) έχει τη χαμηλότερη απώλεια CE (0,86278), γεγονός που υποδηλώνει ότι έχει την καλύτερη επίδοση όσον αφορά την απόδοση των σωστών πιθανοτήτων στις διάφορες κλάσεις. Παρόλο που το μοντέλο 4 (η=0.1, m=0.6) έχει μεγαλύτερη ακρίβεια από το μοντέλο 1 (0.82018 έναντι 0,78959), η απώλεια CE είναι σημαντικά υψηλότερη (3.00950), γεγονός που μας δείχνει ότι οι προβλεπόμενες πιθανότητες είναι συνολικά λιγότερο ακριβείς.  
  
  
  
η=0.001, m=0.2  
Chart, line chart

Description automatically generated  
Chart

Description automatically generated  
  
  
  
η=0.001, m=0.6  
Chart

Description automatically generated  
Chart

Description automatically generated  
  
  
η=0.05, m=0.6  
Chart

Description automatically generated  
Chart, line chart

Description automatically generated  
  
  
η=0.1, m=0.6  
Chart, line chart

Description automatically generated  
  
Chart, line chart

Description automatically generated  
  
  
 ***Α4. Ομαλοποίηση  
(Neural\_Network\_L2.py)***

Η κανονικοποίηση L1 προσθέτει την απόλυτη τιμή των βαρών στη συνάρτηση απώλειας, οδηγώντας σε αραιούς πίνακες βαρών όπου ορισμένα βάρη είναι ακριβώς μηδέν. Η κανονικοποίηση L2, από την άλλη πλευρά, προσθέτει την τετραγωνική τιμή των βαρών στη συνάρτηση απώλειας, η οποία ενθαρρύνει το μοντέλο να κατανέμει τα βάρη πιο ομοιόμορφα μεταξύ των χαρακτηριστικών. Επίσης η κανονικοποίηση L2 είναι καλύτερη όταν υποθέτουμε ότι όλα τα χαρακτηριστικά είναι σχετικά ή συμβάλλουν στην απόδοση του μοντέλου, καθώς τείνει να αποδίδει καλύτερη απόδοση αποτρέποντας τα μεμονωμένα βάρη να γίνουν πολύ μεγάλα. Από την άλλη η κανονικοποίηση L1 είναι προτιμότερη όταν υποψιάζόμαστε ότι ορισμένα χαρακτηριστικά μπορεί να μην είναι σχετικά με το πρόβλημα, καθώς μπορεί να εκτελέσει αποτελεσματικά το feature selection θέτοντας ορισμένα βάρη στο μηδέν.  
  
 Γι αυτό στην περίπτωσή μας θα χρησιμοποιήσουμε την κανονικοποίηση L2.

|  |  |  |  |
| --- | --- | --- | --- |
| Συντελεστής r | CE loss | ΜSE | Acc |
| 0.1 | 1.20814 | 0.86857 | 0.58201 |
| 0.5 | 1.60855 | 3.22297 | 0.30568 |
| 0.9 | 1.60814 | 3.22297 | 0.30568 |

r=0.1  
Chart

Description automatically generated  
Chart

Description automatically generated  
  
  
r=0.5  
Chart

Description automatically generated  
Chart

Description automatically generated  
  
r=0.9  
Chart

Description automatically generated  
Chart

Description automatically generated

***A5. Βαθύ Νευρωνικό Δίκτυο***

***(Deep\_Neural\_Network\_1.py  
Deep\_Neural\_Network\_2.py  
Deep\_Neural\_Network\_3.py  
Deep\_Neural\_Network\_4.py)***  
Για αυτό το ερώτημα θα κρατήσουμε το μοντέλο που τα πήγε καλύτερα στο Α3 ερώτημα με SGD σαν classifier, η=0.001 και m=0.2, και θα κάνουμε εκέι τις παραμετροποιήσεις.

|  |  |  |  |
| --- | --- | --- | --- |
| Περίπτωση | CE loss | ΜSE | Acc |
| 1η | 1.17911 | 0.47850 | 0.81075 |
| 2η | 1.01150 | 0.41434 | 0.84200 |
| 3η | 0.62143 | 0.34520 | 0.80544 |
| 4η | 0.63344 | 0.48102 | 0.86026 |

Για αυτό το ερώτημα θα συμπεριλάβω 4 παραδείγματα με διαφορετικές λογικές:

*1η Περίπτωση:*

* Input layer: 17 nodes
* Hidden layer 1: 34 nodes (increasing)
* Hidden layer 2: 17 nodes (decreasing)
* Output layer: 5 nodes

*Λογική:*  
Σε αυτή την αρχιτεκτονική, το πρώτο κρυφό επίπεδο έχει διπλάσιο αριθμό κόμβων από το επίπεδο εισόδου και το δεύτερο κρυφό επίπεδο έχει τον ίδιο αριθμό κόμβων με το επίπεδο εισόδου. Αυτός ο σχεδιασμός μπορεί να βοηθήσει το δίκτυο να μάθει πιο σύνθετα χαρακτηριστικά επεκτείνοντας την αναπαράσταση στο πρώτο κρυφό επίπεδο και στη συνέχεια συμπυκνώνοντας τις πληροφορίες στο δεύτερο κρυφό επίπεδο.

Chart

Description automatically generated  
Chart

Description automatically generated

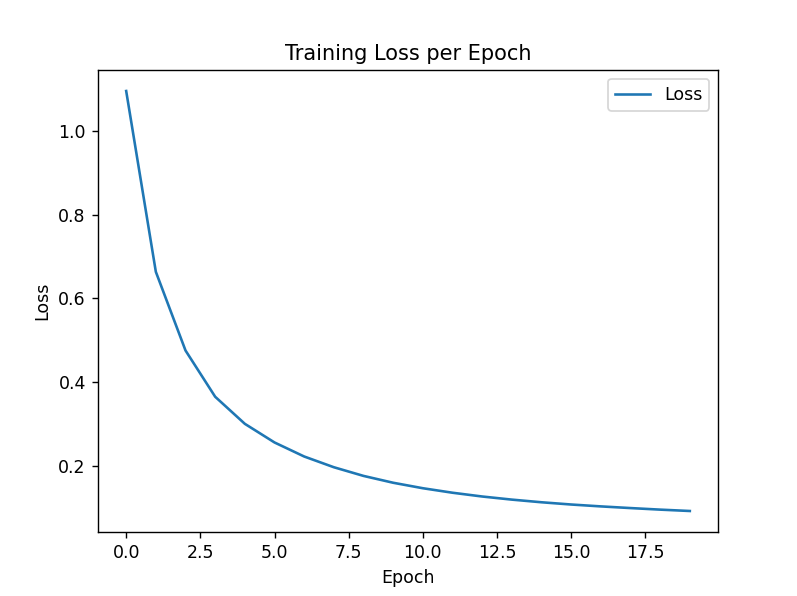
*2η Περίπτωση:*

* Input layer: 17 nodes
* Hidden layer 1: 34 nodes (increasing)
* Hidden layer 2: 17 nodes (decreasing)
* Hidden layer 3: 11 nodes (decreasing)
* Output layer: 5 nodes

*Λογική:*

Αυτή η αρχιτεκτονική είανι η ίδια με πάνω, με τη διαφορά ότι προσθέτουμε ένα κρυφό επίπεδο για να παρατηρήσουμε την συμπεριφορά του μοντέλου.  
Παρατηρούμε ότι η επίδοση του μοντέλου βελτιώθηκε καθώς αυξήθηκε το accuracy και μειώθηκαν τα losses.Chart

Description automatically generated



3η Περίπτωση:

* Input layer: 17 nodes
* Hidden layer 1: 12 nodes (decreasing)
* Hidden layer 2: 8 nodes (decreasing)
* Hidden layer 3: 4 nodes (decreasing)
* Output layer: 5 nodes

*Λογική:*Σε αυτή την αρχιτεκτονική, ο αριθμός των κόμβων σε κάθε κρυφό στρώμα μειώνεται προοδευτικά. Αυτός ο σχεδιασμός μπορεί να βοηθήσει το δίκτυο να μάθει μια ιεραρχική αναπαράσταση των δεδομένων, όπου κάθε επίπεδο συλλαμβάνει προοδευτικά πιο αφηρημένα χαρακτηριστικά.Chart, line chart

Description automatically generated

Chart

Description automatically generated

4η Περίπτωση:

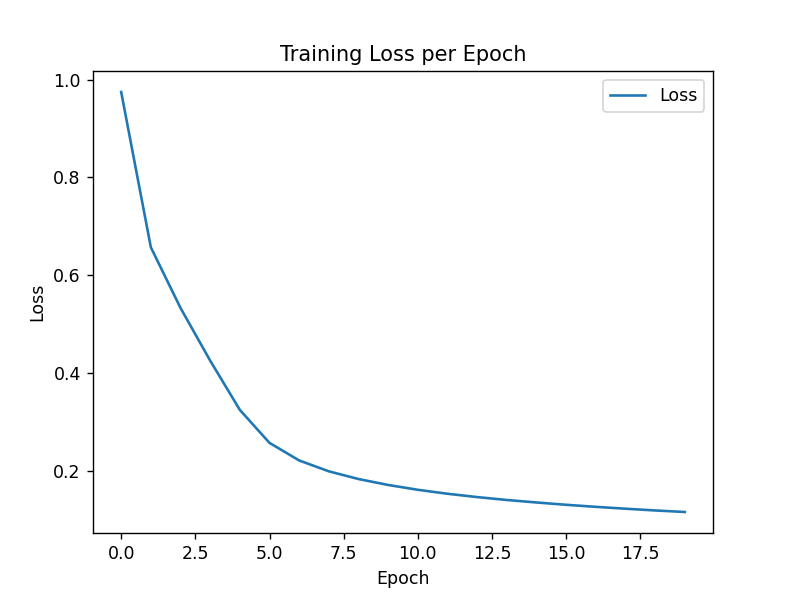
* Input layer: 17 nodes
* Hidden layer 1: 20 nodes (increasing)
* Hidden layer 2: 20 nodes (same)
* Hidden layer 3: 10 nodes (decreasing)
* Output layer: 5 nodes

Λογική:

Σε αυτή την αρχιτεκτονική, το πρώτο και το δεύτερο κρυφό επίπεδο έχουν τον ίδιο αριθμό κόμβων, επιτρέποντας στο δίκτυο να μαθαίνει παρόμοια levels of abstraction σε αυτά τα επίπεδα. Το τρίτο κρυφό επίπεδο έχει λιγότερους κόμβους, γεγονός που μπορεί να βοηθήσει στη συμπύκνωση των πληροφοριών και στη μείωση της διαστατικότητας των δεδομένων πριν από το επίπεδο εξόδου.

Chart

Description automatically generated



**Συμπέρασμα:**Θεωρώ πως το καλύτερο μοντέλο από τα 4 είναι αυτό της 4ης περίπτωσης καθώς συνδυαστικά έχει το καλύτερο accuracy και το δεύτερο μικρότερο CE loss το οποίο συναγωνίζεται με αυτό της της 3ης περίπτωσης με το συνολικά μικρότερο CE loss.  
Άρα για το συγκεκριμένο πρόβλημα, καλύτερη απόδοση είχαν τα μοντέλα με 3 layers και μείωση ή σχετικά μη (μεγάλη) αύξηση κόμβων στο 1ο κρυφό επίπεδο.