ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

Ακαδημαϊκό έτος: 2020-2021 Εαρινό Εξάμηνο

Ονοματεπώνυμο: Πατάκης Ανδρέας
Αριθμός Μητρώου: Π17103

Τεχνητή Νοημοσύνη και Έμπειρα Συστήματα
Εργασία 2

Σκέλος 2.2:

Νευρωνικό Δίκτυο

Βασικές πληροφορίες

Το νευρωνικό δίκτυο το οποίο δημιουργήθηκε προσπαθεί να κάνει των διαχωρισμό μεταξύ των γραμμάτων “Α” και “Π”. Το σύνολο δεδομένων το οποίο χρησιμοποιήθηκε, πάρθηκε απο ένα dataset της σελίδα που μας δώσατε στην εκφώνηση http://www.ics.uci.edu/~mlearn/MLRepository.html. Συγκεκριμένα το dataset το οποίο χρησιμοποιήθηκε μπορεί να το βρεθεί στον παρακάτω σύνδεσμο: https://archive.ics.uci.edu/ml/datasets/Letter+Recognition

Dataset

Το συγκεκριμένο dataset αρχικά περιέχει δεδομένα για όλα τα γράμματα της αγγλικής αλφαβήτου. Συνολικά διαθέτει 20.000 δεδομένα. Απο την στιγμή όμως που εμείς θέλουμε να κάνουμε τον διαχωρισμό μεταξύ των γραμμάτων “Α” και “Π” αρκεί να χρησιμοποιήσουμε μονάχα τα διαθέσιμα δεδομένα για αυτά τα δύο. Αυτο σημαίνει οτι τελικό σύνολο δεδομένων το οποίο διαθέτουμε κατέρχεται στα 1.592 δεδομένα με τις ακόλουθες αντιστοιχίες :

- Α: 789
- Π: 803

Κάθε ένα απο τα παραπάνω δεδομένα περιέχει στοιχεία για 16 διαφορετικές παραμέτρους οι οποίες αφορούν το σχήμα του γράμματος. Αυτό σημαίνει πως έχουμε 16 input features για το νευρωνικό μας δίκτυο. Τα features αυτά βάση το documentation του παραπάνω dataset είναι τα εξής:

- x-box horizontal position of box (integer)
- y-box vertical position of box (integer)
- width width of box (integer)
- high height of box (integer)
- onpix total # on pixels (integer)
- x-bar mean x of on pixels in box (integer)
- y-bar mean y of on pixels in box (integer)
- x2bar mean x variance (integer)
- y2bar mean y variance (integer)
- xybar mean x y correlation (integer)
- x2ybr mean of x * x * y (integer)
- xy2br mean of x * y * y (integer)
- x-ege mean edge count left to right (integer)
- xegvy correlation of x-ege with y (integer)
- y-ege mean edge count bottom to top (integer)
- yegvx correlation of y-ege with x (integer)

Νευρωνικό Δίκτυο

Για το νευρωνικό δίκτυο δεν χρησιμοποιήθηκε κάποια βιβλιοθήκη αλλα έγινε προσωπική υλοποίηση. Δημιουργήθηκε ενα πακέτο το οποίο καλείτε και με πολύ λίγες γραμμές κώδικα το νευρωνικό ξεκινάει την λειτουργία του. Κατά την δημιουργία του θέτουμε τον αριθμό των input features, των στρωμάτων, των νευρώνων ανά στρώμα και του learning rate.
Π.χ:

```
net = Neural_Network(input_features,[10,4,2],learning_rate)
```

Για την προπόνηση του νευρωνικού δικτύου χρησιμοποιείται μια συνάρτηση train() και για τον έλεγχο του μια συνάρτηση test() αντίστοιχα.

- Η συνάρτηση train() δέχεται σαν ορίσματα τα set δεδομένων (input features) για την εκπαίδευση, το αποτέλεσμα κάθε set (observed value ή y), των αριθμών των εποχών για τις οποίες θέλουμε να τρέξει τα δεδομένα το νευρωνικό δίκτυο καθώς και το batch size το οποίο επιθυμούμε.
- Η συνάρτηση test() δέχεται σαν ορίσματα τα set δεδομένων (input features) τα οποία θα χρησιμοποιήσουμε για να ελέγξουμε την ακρίβεια του ταξινομητή μας, και τις αντίστοιχες τιμές αποτελέσματα για κάθε set (observed values)

Π.χ:

```
net.train(training_set,training_output,epochs,batch_size)
net.test(testing_set,testing_output)
```

Ο αλγόριθμος σύγκλισης που χρησιμοποιεί το νευρωνικό δίκτυο είναι ο Gradient Descent και σαν activation function χρησιμοποιείται η σιγμοειδής καμπύλη σε όλες τους νευρώνες,

Κατανομή και 10-fold cross validation

Προκειμένου να αποκτήσουμε μια πιο αντικειμενική και όσο το δυνατόν λιγότερο μεροληπτική αντιμετώπιση απέναντι στο πρόβλημα που έχουμε να λύσουμε χρησιμοποιήθηκαν 2 βασικές ιδέες.

Η πρώτη αφορά την προσπάθεια να πετύχουμε ομοιόμορφη κατανομή των δεδομένων στο dataset το οποίο χρησιμοποιούμε. Για παράδειγμα στο πρόβλημα μας αν το dataset ήταν χωρισμένο ακριβώς στα δύο, με τα “Α” να προηγούνται των “Π” έως ότου τελειώσουν, τότε το νευρωνικό μας δίκτυο θα έδειχνε ξεκάθαρη προτίμηση στα “Α” με αποτέλεσμα να παίρνουμε ταξινομητική ακρίβεια περίπου 50% αφού θα ταξινομούσε μονο τα μισά δεδομένα σωστά. Για αυτό τον λόγο αφού φορτώθηκε το dataset και έγινε η απαραίτητη εκκαθάριση, κρατώντας μονάχα τα γράμματα τα οποία μας ενδιαφέρουν, έγινε ενα τυχαίο ανακάτεμα (shuffle) προκειμένου να προσπαθήσουμε να προσομοιάσουμε μια ομοιόμορφη κατανομή δεδομένων. Τέλος τα αποτελέσματα φορτώθηκαν σε 2 αρχεία csv, τα letters_x και letters_y, με το letters_x να περιέχει τα 16 input features για κάθε ένα απο τα 1.592 δεδομένα μας και το letters_y να περιέχει το αποτέλεσμα καθενός απο τα παραπάνω 1.592 αποτελέσματα(αν είναι Α ή Π δηλαδή). Στο παράδειγμα μας τα Α έχουν την αναπαράσταση [1,0] ενώ τα Π την [0,1].

Επιπλέον χρησιμοποιήθηκε η τεχνική του 10-fold cross validation προκειμένου να καταφέρουμε να χρησιμοποιήσουμε το νευρωνικό δίκτυο του οποίου τα δεδομένα φαίνεται να καταφέρνουν να κάνουν την πιο ακριβή ταξινόμηση. Αυτό σημαίνει πως τα 1.592 χωρίστηκαν με 10 διαφορετικούς τρόπους (όλες οι πιθανές αναθέσεις), αφήνοντας κάθε φορά 1.431 δεδομένα εκπαίδευσης και 159 δεδομένα για έλεγχο(test).

Σύνοψη

Τελικά, εφαρμόζοντας τις παραπάνω τεχνικές καταφέραμε να δημιουργήσουμε ένα νευρωνικό δίκτυο με ταξινομητική ακρίβεια κοντά στο 97%. Προκειμένου να φτάσουμε σε αυτό το ποσοστό χρησιμοποιήθηκαν οι εξής παράμετροι:

- 3 στρώσεις με 10, 4, 2 νευρώνες αντίστοιχα.
- Learning rate: 0.02
- Εποχές: 10
- Batch size: 40

Αφού εξεταστούν και τα 10 folds, κρατάμε το νευρωνικό δίκτυο το οποίο φαίνεται να έχει την μεγαλύτερη ταξινομητική ακρίβεια με βάση το testing accuracy το οποίο επέστρεψε.

Τέλος τυπώνουμε στον χρήστη και τις 159 προσπάθειες ταξινόμησης(του καλύτερου fold), δείχνοντας ποια γράμματα ταξινομήθηκαν σωστά και ποια λανθασμένα.

Ακολουθούν στιγμιότυπα εκτέλεσης

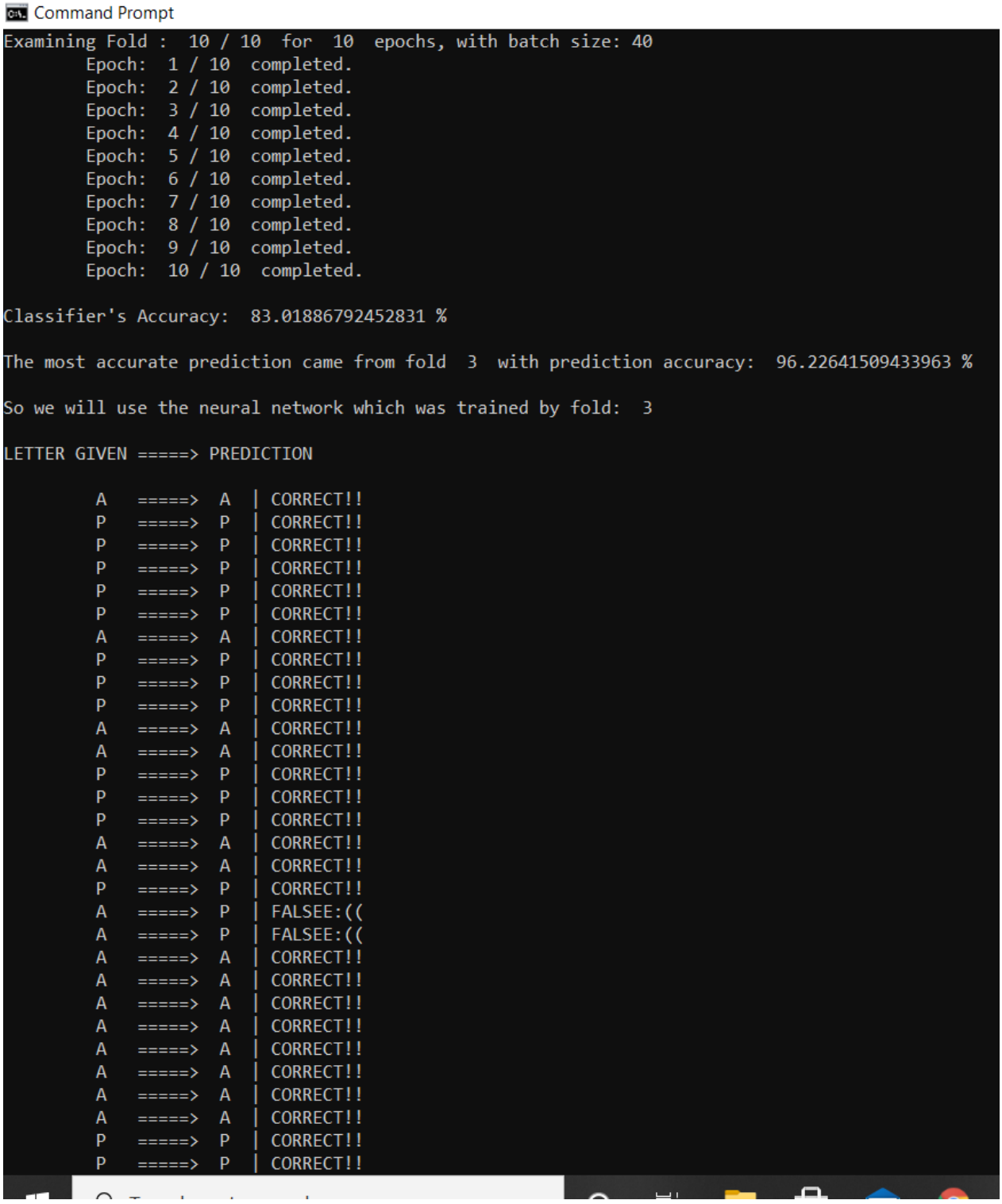
Screenshots

Όσο εξετάζονται τα folds

Command Prompt

```
Examining Fold :  3 / 10  for  10  epochs, with batch size: 40
        Epoch:  1 / 10  completed.
        Epoch:  2 / 10  completed.
        Epoch:  3 / 10  completed.
        Epoch:  4 / 10  completed.
        Epoch:  5 / 10  completed.
        Epoch:  6 / 10  completed.
        Epoch:  7 / 10  completed.
        Epoch:  8 / 10  completed.
        Epoch:  9 / 10  completed.
        Epoch:  10 / 10  completed.

Classifier's Accuracy:  96.22641509433963 %


Examining Fold :  4 / 10  for  10  epochs, with batch size: 40
        Epoch:  1 / 10  completed.
        Epoch:  2 / 10  completed.
        Epoch:  3 / 10  completed.
        Epoch:  4 / 10  completed.
        Epoch:  5 / 10  completed.
        Epoch:  6 / 10  completed.
        Epoch:  7 / 10  completed.
        Epoch:  8 / 10  completed.
        Epoch:  9 / 10  completed.
        Epoch:  10 / 10  completed.

Classifier's Accuracy:  92.45283018867924 %


Examining Fold :  5 / 10  for  10  epochs, with batch size: 40
        Epoch:  1 / 10  completed.
        Epoch:  2 / 10  completed.
        Epoch:  3 / 10  completed.
        Epoch:  4 / 10  completed.
        Epoch:  5 / 10  completed.
        Epoch:  6 / 10  completed.
        Epoch:  7 / 10  completed.
        Epoch:  8 / 10  completed.
        Epoch:  9 / 10  completed.
        Epoch:  10 / 10  completed.

Classifier's Accuracy:  87.42138364779875 %


Examining Fold :  6 / 10  for  10  epochs, with batch size: 40
        Epoch:  1 / 10  completed.
        Epoch:  2 / 10  completed.
```

Fold το οποίο επιλέχθηκε τελικά και παρουσίαση σωστών και λανθασμένων ταξινομήσεων



Αποτελέσματα και τελικό score σωστών ταξινομήσεων