2η εργασια

ελενη τζαρα-Π18151 /αντωνιοσ ελευθεριοσ καρναβασ-Π18063

ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ και εμπειρα συστηματα

1. Γενετικός αλγόριθμος

Η εργασία υλοποιήθηκε με τη χρήση της προγραμματιστικής γλώσσας **python**.

Η **λογική**, αυτής της υλοποίησης του γενετικού αλγορίθμου έχει ως εξής:

Ο χρήστης επιλέγει ποιο γράμμα επιθυμεί να του εμφανιστεί, και αναλόγως, εκχωρείται στην μεταβλητή **letter** ένα string 77 χαρακτήρων (11x7=77) το οποίο αποτελείται από 1 και 0, και αν το ανάγουμε σε έναν πίνακα 11x7 θα δώσει το ζητούμενο αποτέλεσμα.

Για παράδειγμα το γράμμα Κ στο string αναπαρίσταται ως 10000000000010100000000010001000000010000010000000001000100000000000101000000 και κατ’ επέκταση σε αυτόν τον 11x7 πίνακα

1 0 0 0 0 0 0

0 0 0 0 0 1 0

1 0 0 0 0 0 0

0 0 0 1 0 0 0

1 0 0 0 0 0 0

0 1 0 0 0 0 0

1 0 0 0 0 0 0

0 0 0 1 0 0 0

1 0 0 0 0 0 0

0 0 0 0 0 1 0

1 0 0 0 0 0 0

Αρχικά δηλώνουμε τις εξής απαραίτητες μεταβλητές:

**letter**: το string «στόχος» που θέλουμε να εμφανιστεί.

**pop:** ο αριθμός του πληθυσμού.

**genes:** τα αποδεκτά «γονίδια» που θα δομήσουν ένα «χρωμόσωμα»(0 και 1)

**t:** αν ο χρήστης πληκτρολόγησε Κ ή Τ η μεταβλητή παίρνει την τιμή True

Χρησιμοποιήσαμε επίσης και δύο πακέτα, το **random** προκειμένου να προσθέσουμε τυχαιότητα στη διαδικασία, και το **tkinter** προκειμένου να εμφανίσουμε το επιθυμητό grid με το τελικό αποτέλεσμα.

Text

Description automatically generated

Δημιουργούμε μία κλάση **Individual(object)**, η οποία θα αναπαριστά κάθε μεμονωμένο «άτομο» του πληθυσμού.

Text

Description automatically generated

Μέσα σε αυτή εκτός από τον constructor, def \_\_init\_\_(self,chromosome), υπάρχουν οι εξής μέθοδοι:

* Def **mutated\_genes(self)**

Η μέθοδος αυτή επιστρέφει μέσω της μεταβλητής **gene**, ένα από τα δύο εφικτά «γονίδια» , τυχαία, είτε το 0 είτε το 1.

* Def **create\_gnome(self)**

Η μέθοδος αυτή επιστρέφει ένα string ίδιου μήκους με το **letter**, το οποίο αποτελείται από 0 και 1 τα όποια έχουν τοποθετηθεί με τυχαία σειρά, με τη βοήθεια της μεθόδου mutated\_genes.

* Def **mate(self,par2)**

Με αυτή τη μέθοδο πραγματοποιούμε την «αναπαραγωγή». Αρχικοποίουμε την μεταβλητή **child\_chromosome**, και στη συνέχεια με τη χρήση μιας επανάληψης, παίρνουμε κάθε «γονίδιο»( στοιχείο )από το string του γονέα 1(self) και του string του γονέα 2 (par2), και με βάση τη τυχαία πιθανότητα που δημιουργούμε σε κάθε επανάληψη, εκχωρούμε στο child\_chromosome είτε το γονίδιο του γονέα 1, είτε του γονέα 2, είτε ένα τυχαίο(0 ή 1) που θεωρείται το «μεταλλαγμένο».

Text

Description automatically generated

* Def **calc\_fitness(self)**

Η μέθοδος αυτή επιστρέφει μέσω της μεταβλητής **fitness** , την βαθμολογία που προκύπτει από την συνάρτηση καταλληλόλητας. Στην υλοποίηση αυτή το fitness score δείχνει πόσες διαφορές υπάρχουν ανάμεσα στο string του τρέχον εξεταζόμενου και του string του «στόχου» (letter) .

* Def **grid\_proccess(str, k)** και **grid\_like(str)**

Χρησιμοποιούνται για την προβολή των αποτελεσμάτων.

Text

Description automatically generated

**Driver code**

* Αρχικοποιούμε την μεταβλητή **generation** ίση με 1 αφού αναφερόμαστε στην πρώτη γενιά, θέτουμε στην μεταβλητή **found** την τιμή False και αρχικοποιούμε την λίστα **population**.

Αν ο χρήστης έχει δώσει έγκυρο input( Κ ή Τ)

* Δημιουργούμε τον αρχικό πληθυσμό, χρησιμοποιώντας την μέθοδο **create\_gnome()**, και εκχωρώντας τα αποτελέσματα στην **population**.

Όσο η μεταβλητή **found** παραμένει False:

* Ταξινομούμε την **population** σε αύξουσα σειρά, με βάση το **fitness** του κάθε στοιχείου.
* Αν το στοιχείο με το χαμηλότερο fitness, δηλαδή το πρώτο στοιχείο, έχει **fitness**=0 τότε εκχωρείται στο **found** η τιμή True και η επανάληψη σταματάει.

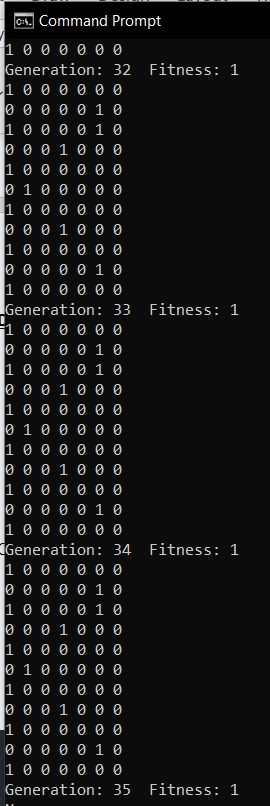


* Αλλιώς, ξεκινάει η διαδικασία για την ανανέωση του πληθυσμού και αρχικοποιούμε την λίστα **new\_generation**.
* Πραγματοποιώ μερική ανανέωση πληθυσμού, μεταφέροντας το 10% των στοιχείων με το καλύτερο fitness απευθείας στην επόμενη γενιά.
* Έπειτα, στον υπόλοιπο πληθυσμό(90%) πραγματοποιώ αναπαραγωγή με τη χρήση της μεθόδου **mate,** και προσθέτω το αποτέλεσμα(offspring) στην **new\_generation**.
* Ανανεώνω την **population,** εκχωρώντας της τη **new\_generation ,** εκτυπώνω το καταλληλότερο κατά το fitness string που έχει διαμορφωθεί μαζί με τον αριθμό της γενιάς και το τρέχον fitness,και πριν το τέλος του loop, αυξάνω το generation κατά 1.

Text

Description automatically generated

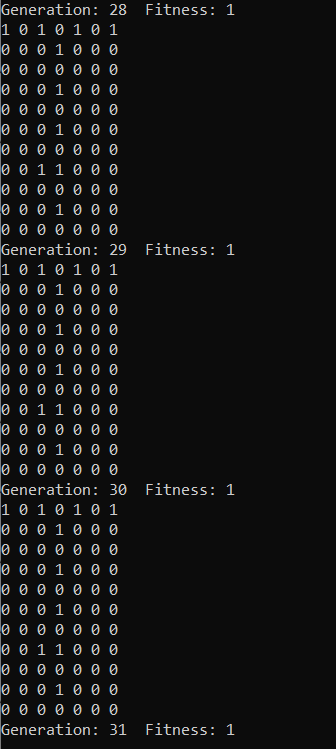
**Αποτέλεσμα για το γράμμα Κ**



A screenshot of a computer

Description automatically generated with medium confidence

**Αποτέλεσμα για το γράμμα T**



A screenshot of a computer

Description automatically generated with low confidence

1. Νευρωνικό δίκτυο

Η 2η άσκηση υλοποιήθηκε σε γλώσσα python και το νευρωνικό δίκτυο δημιουργήθηκε με 2 τρόπους, με την χρήση της βιβλιοθήκης TensorFlow και με χειροκίνητο τρόπο.

Αρχικά τα δεδομένα συλλέχθηκαν από το παρακάτω dataset, το link του οποίου είναι: <http://archive.ics.uci.edu/ml/datasets/Letter+Recognition>

Το dataset περιέχει 20000 χειρόγραφα από όλα τα κεφαλαία γράμματα του λατινικού αλφάβητου. Κάθε γράμμα δεν είναι σε μορφή φωτογραφίας του χειρόγραφου αλλά έχει γίνει ήδη κάποια προ επεξεργασία με αποτέλεσμα το κάθε γράμμα να προσδιορίζεται από ένα διάνυσμα 16 τιμών που κάθε τιμή έχει κάποια αριθμητική πληροφορία που χαρακτηρίζει το κάθε γράμμα. Οι τιμές αυτές δηλώνουν τα παρακάτω:

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

Και ένα παράδειγμα από τα δεδομένα είναι:

T,2,8,3,5,1,8,13,0,6,6,10,8,0,8,0,8

I,5,12,3,7,2,10,5,5,4,13,3,9,2,8,4,10

D,4,11,6,8,6,10,6,2,6,10,3,7,3,7,3,9

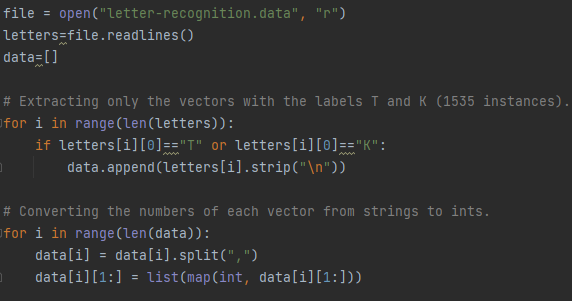
N,7,11,6,6,3,5,9,4,6,4,4,10,6,10,2,8

G,2,1,3,1,1,8,6,6,6,6,5,9,1,7,5,10

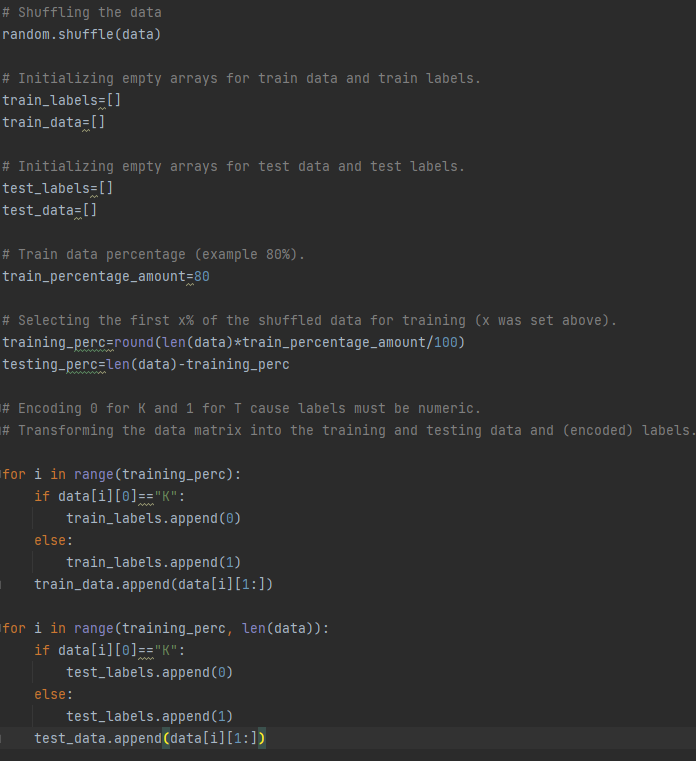
S,4,11,5,8,3,8,8,6,9,5,6,6,0,8,9,7

Εμείς όμως θα φτιάξουμε 2 classifier μόνο για την αναγνώριση μεταξύ των γραμμάτων Κ και T. Το σύνολο αυτών των γραμμάτων που έχουμε δεδομένα είναι 1535.

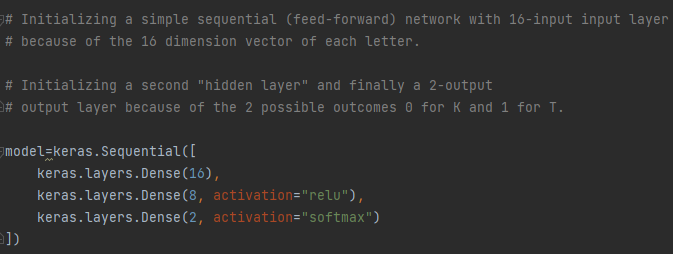
Αρχικά κάνουμε μια επεξεργασία των δεδομένων για να τα φέρουμε στη μορφή που επιθυμούμε με τον παρακάτω τρόπο:



‘Έπειτα ανακατεύουμε τα δεδομένα, δηλώνουμε το ποσοστό των δεδομένων που θέλουμε για training (80% as default) και κάνουμε encode τα Κ και Τ να τα αναγνωρίζουμε ως 0 και 1 αντίστοιχα. Τέλος σπάμε τα δεδομένα σε training data και testing data με τα αντίστοιχα labels(0 για Κ και 1 για Τ) που χαρακτηρίζουν την κάθε πλειάδα ανάλογα με το ποσοστό training(80%) που είχαμε ορίσει.



1η υλοποίηση (TensorFlow-Keras)

Αρχικοποιούμε το νευρωνικό μας δίκτυο με input layer ίσο με 16 όσα και τα χαρακτηριστικά του κάθε γράμματος δηλαδή, ένα hidden layer με 8 εισόδους και συνάρτηση ενεργοποίησης την relu και τέλος ένα output layer 2 εξόδων αφού τα πιθανά αποτελέσματα είναι 0 για Κ ή 1 για Τ.

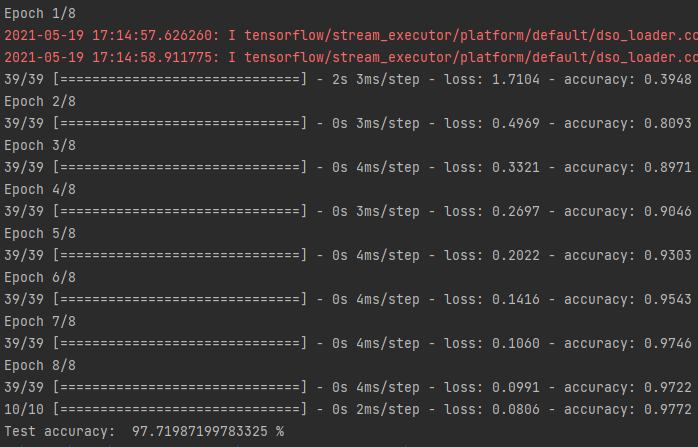
Ύστερα βάζουμε εμείς κάποιες παραμέτρους, στη συγκεκριμένη περίπτωση τον optimizer «Adam», την συνάρτηση κόστους «sparse\_categorical\_crossentropy» και το αποτέλεσμα που θέλουμε δηλαδή την ακρίβεια ταξινόμησης.

Έπειτα κάνουμε train το δίκτυο μας με το training dataset που ορίσαμε παραπάνω σε 8 εποχές.

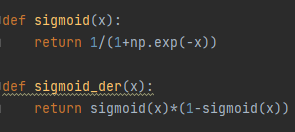


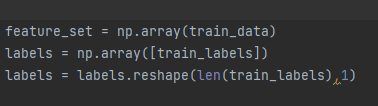
Τέλος κάνουμε test το μοντέλο με το testing dataset που ορίσαμε και παίρνουμε μια ακρίβεια συνήθως της τάξης του 90%>.

Παράδειγμα εκτέλεσης του κώδικα:



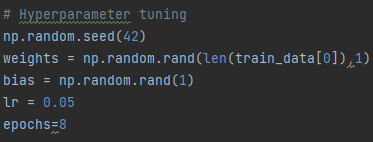
2η υλοποίηση(Χειροκίνητα)

Αρχικά ορίζω την απαραίτητη Σιγμοειδή συνάρτηση καθώς και την παράγωγο της Σιγμοειδούς.

Έπειτα ορίζω τα χαρακτηριστικά των γραμμάτων και τα αντίστοιχα labels τους οργανωμένα σε arrays με την βοήθεια της βιβλιοθήκης NumPy.

Ύστερα ορίζω κάποιες απαραίτητες παραμέτρους οι οποίες είναι:

* μια εντολή για τυχαίους αριθμούς κάθε φορά που υλοποιείται το πρόγραμμα
* τυχαίος πίνακας βαρών με διάσταση x όσα και τα χαρακτηριστικά κάθε γράμματος
* τυχαία τιμή του bias term
* learning rate ίσο με 0.5 (μπορεί να αλλαχθεί)
* 8 εποχές όσες και την 1η υλοποίηση έτσι ώστε να συγκρίνουμε αποτελέσματα ακρίβειας

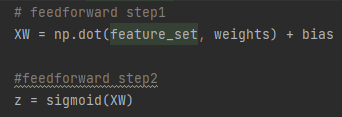


Τώρα υλοποιούμε τον αλγόριθμο mean squared error έτσι ώστε να υπολογίσουμε επαναληπτικά τον πίνακα των βαρών.

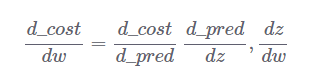
Ο τύπος του MSE, δηλαδή του error(cost) είναι ο παρακάτω:



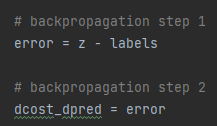
Δηλαδή ο τύπος που θα χρησιμοποιούμε κατά την κανονική διάσχιση (feed-forward) του δικτύου.



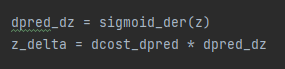
Εμείς πρέπει να σε κάθε εποχή στην αντίθετη διάσχιση (back propagation) να υπολογίσουμε το παρακάτω:



Το πρώτο κλάσμα είναι στην ουσία το σφάλμα predicted-observed δηλαδή η διαφορά των labels με την εκάστοτε πρόβλεψη.



Για το δεύτερο κλάσμα απλά υπολογίζουμε την παράγωγο-σιγμοειδή των predicted τιμών. Τέλος πολλαπλασιάζουμε τα 2 παραπάνω κλάσματα.



Τέλος πρέπει να πολλαπλασιάσουμε το d\_cost/dw που βρήκαμε με τα διανύσματα των γραμμάτων. Για να γίνει αυτό υπολογίζουμε τον ανάστροφο του πίνακα διανυσμάτων για να γίνει σωστά ο πολλαπλασιασμός πινάκων. Βάζουμε στον πολλαπλασιασμό και το learning rate και το αφαιρούμε από τον πίνακα των βαρών (που είτε υπάρχει ήδη από προηγουμένη εποχή, είτε δημιουργήθηκε τυχαία στην πρώτη εποχή).

Για το testing υπολογίζουμε την ακρίβεια η οποία με 8 εποχές είναι περίπου πάνω από 55%. Αισθητά χαμηλότερη ακρίβεια από το νευρωνικό δίκτυο με τη χρήση TensorFlow-keras.