

Νευρωνικά Δίκτυα (Πρώτο Παραδοτέο)

PaRT 1. NEAREST CENTROID & knn IN CYFAR 10

PaRT 2. MULTI LAYER PERCEPTRONS IN CYFAR 10

Καλλιμάνης Ιωάννη | 10007 | Ηλεκτρολόγων μηχανικών και μηχανικών ηλεκτρονικών υπολογιστών | [ikallima@ece.auth.gr](mailto:ikallima@ece.auth.gr) | 26-11-2023

PART 1.

Εισαγωγή

Ξεκινώντας την ενασχόληση με την εργασία, αποφάσισα να ασχοληθώ με το database της CYFAR-10 με το πρώτο μέρος να πραγματεύεται την απόδοση που μπορούν να δώσουν απλοί αλγόριθμοι μηχανικής μάθησης σε ένα τόσο μεγάλο πρόβλημα classification.

Περιγραφή κώδικα

Στον κώδικα μου λοιπόν, αξιοποιώντας την συνάρτηση unpickle που μου την παρείχε έτοιμη η ιστοσελίδα της CYFAR-10 για να «φορτώσω» τα δεδομένα μου από τα εκάστοτε mini-batch που είναι αποθηκευμένα. Στην συνέχεια ορίζω της συναρτήσεις που για nearest centroid και k-nearest-neighbors και μια συνάρτηση που επιστρέφει τις μετρικές του κάθε μοντέλου (accuracy, f1\_score, recall). Στην συνέχεια φθάνουμε στην «main» του κώδικα μου όπου σώζω τα δεδομένα μου σε 4 πίνακες (2 για την διαδικασία εκπαίδευσης (δεδομένα και κλάση) και 2 για την διαδικασία εκτίμησης). Εκπαιδεύω στην συνέχεια το καθένα από τα μοντέλα μου με βάση τα δεδομένα εκπαίδευσης και τεστάρω την απόδοση τους εκτυπώνοντας τις μετρικές που αναφέρθηκαν παραπάνω. Τα αποτελέσματα που πείρα είναι:

Centroid Accuracy: 27.74%

Centroid F1 Score: 0.2541

Centroid Recall: 0.2774

1-NN Accuracy: 35.39%

1-NN F1 Score: 0.3495

1-NN Recall: 0.3539

3-NN Accuracy: 33.03%

3-NN F1 Score: 0.3192

3-NN Recall: 0.3303

Να προσθέσω, επίσης, καθώς είχα δοκιμάσει τον κώδικα μου ώστε να δω τα αποτελέ-

σματα του, λόγω των «χαμηλών» μετρικών πρόσθεσα μέσω της pca ένα απλουστευμένο

2D διάγραμμα ώστε να δω πως είναι τα δεδομένα μου στον δισδιάστατο χώρο.(σχ. 1)

Συμπεράσματα

Από το διάγραμμα αυτό, διαπίστωσα ότι τα δεδομένα μου είναι όλα λίγο-πολύ κοντά μεταξύ τους και συγκεντρωμένα σε μια περιοχή, οπότε είναι αρκετά δύσκολο με αλγόριθμους μηχανικής μάθησης σαν τους 2 που χρησιμοποιήθηκαν να πάρουμε ικανοποιητικά υψηλές μετρικές

Σημειώσεις

Ο χρόνος εκτέλεσης του κώδικα μου ανέρχεται στα 5-7 λεπτά. Για προσωπικούς λόγους έτρεξα τον κώδικα μέχρι k = 17 nearest neighbors (ουσιαστικά τους 7 πρώτους πρώτους αριθμούς εξαιρώντας το 2 [1 3 5 7 11 13 17], χρησιμοποιήθηκαν πρώτοι αριθμοί για να έχω πάντα μια επικρατούσα κλάση στους μεγάλους αριθμούς, και 7 καθώς με 1 ακόμα από την centroid να τρέξουν με multi-processing (ή και multithreading) στους 8 πυρήνες του λάπτοπ μου) και διαπίστωσα τουλάχιστον στην μετρική accuracy το καλύτερο μοντέλο να είναι αυτό με k = 1, με μικρές διαφορές της τάξης έως 3-4% από κάθε μοντέλο από τα υπόλοιπα. Καθώς όμως καθυστερούσε αρκετά παραπάνω το πρόγραμμα μου προτίμησα να μην τα συμπεριλάβω στο τελικό παραδοτέο μου.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, πολυχρωμία

Περιγραφή που δημιουργήθηκε αυτόματα(σχ. 1)

**PART 2.**

Εισαγωγή

Στην συνέχεια, λοιπόν, μου ζητήθηκε να ασχοληθώ με το πρόβλημα της ταξινόμησης του παραπάνω μοντέλου με την χρήση MLP (MULTI LAYER PERCEPTRON). Αποφάσισα και εγώ με την σειρά μου να αφιερώσω το 2ο κομμάτι της εργασίας μου σε δημιουργία MLP from scratch, χωρίς την χρήση έτοιμου λογισμικού. Στόχος μου ήταν να κατανοήσω βαθύτερα την θεωρία που κρύβεται πίσω από τους MLP και να διαπιστώσω ποια είναι τα προβλήματα που θα συναντήσω από την θεωρεία στην πράξη και πως θα τα λύσω. Σαφώς με την χρήση frameworks τύπου pytorch η δουλεία μου θα γινόταν τόσο πιο γρήγορη, όσο και πιο αποτελεσματική, απλά προτίμησα για εκπαιδευτικούς λόγους αυτήν την στρατηγική.

Περιγραφή κώδικα

Καθώς η φιλοσοφία του κώδικα μου ήταν η δημιουργία MLP from scratch οι μόνες βιβλιοθήκες που χρειάστηκα να κάνω import είναι η numpy και η matplotlib. Ξεκινώντας την περιγραφή του μοντέλου, όπως και παραπάνω εισήγαγα τα δεδομένα μου στον κώδικα, τα χώρισα σε train\_data, train\_labels, test\_data και test\_labels (όπου ό,τι αναφέρεται σε train είναι τα δεδομένα εκπαίδευσης του μοντέλου και test τα δεδομένα εκτίμησης του μοντέλου, ενώ ό,τι αναφέρεται σε data είναι οι πίνακες των δεδομένων με γραμμές όλα τα παραδείγματα και στήλες όλα τα χαρακτηριστικά του κάθε παραδείγματος, ενώ labels είναι τα διανύσματα που αποθηκεύουν στην θέση i σε ποια κλάση ανήκει το παράδειγμα i). Στην συνέχεια, καλώ την συνάρτηση gradient\_descent όπου παίρνει σαν εισόδους τα δεδομένα εκπαίδευσης και εκτίμησης με τις ετικέτες τους καθώς και άλλα χαρακτηριστικά που αφορούν το MLP που δημιουργείται, όπως οι αριθμοί των νευρώνων σε κάθε επίπεδο, για πόσες εποχές θα τρέξει ο κώδικας, ανά πόσες εποχές θα εκτυπώνετε το αποτέλεσμα, το μέγεθος του κάθε minibatch που θα κάνουμε ενημέρωση των βαρών καθώς και το βήμα εκπαίδευσης. Αφού καλέσουμε την συνάρτηση με τα ορίσματα που επιθυμούμε γίνεται μία κανονικοποίηση τόσο του training set όσο και του test set για να ξεκινήσει ο κώδικας μας (η κανονικοποίση γίνεται με τέτοιον τρόπο ώστε κάθε χαρακτηριστικό ξεχωριστά στο σύνολο των παραδειγμάτων εκπαίδευσης να έχει μέση τιμή μηδέν και τυπική διασπορά ίση με την μονάδα. Για τον λόγο αυτό χρησιμοποιήθηκαν οι συναρτήσεις np.mean και np.std στα δεδομένα του πίνακα εκπαίδευσης, αλλά τα αποτελέσματα τους για την διαδικασία αυτή χρησιμοποιήθηκαν τόσο στην κανονικοποίηση του ίδιου πίνακα εκπαίδευσης αλλά και του πίνακα εκτίμησης). Στην συνέχεια αρχικοποιούμε τα βάρη και τα bias των Perceptron για να προχωρήσουμε από gaussian κατανομή με μέση τιμή μηδέν και τυπική διασπορά ίση με μονάδα, στο κύριο μέρος του κώδικας μας, την εκπαίδευση των Perceptron μας όπου κάθε «εποχή» εκτελούνται τα παρακάτω:

Forward propagation: η συνάρτηση αυτή χρησιμοποιείται τόσο από τα δεδομένα εκπαίδευσης όσο και από τα δεδομένα εκτίμησης. Αποτελεί την έξοδο που παίρνουμε από το μοντέλο μας. Στην διαδικασία της εκπαίδευσης ένα minibatch των δεδομένων πολλαπλασιάζεται στο πρώτο επίπεδο με τα αντίστοιχα βάρη του κάθε νευρώνα και προστίθεται και το bias του κάθε νευρώνα ώστε να πάρουμε ως έξοδο ένα αριθμό για κάθε νευρώνα και για κάθε δείγμα του minibatch ώστε να περάσει ο κάθε αριθμός αυτός από την συνάρτηση ενεργοποίησης του επιπέδου εισόδου (στην περίπτωση μας την relu) και να μας δώσει τελικά την έξοδο του νευρώνα, η οποία κανονικοποιείται με μέση τιμή το μηδέν και τυπική διασπορά ίση με μονάδα πριν μπουν ως δεδομένα εισόδου στο επόμενο επίπεδο του νευρωνικού μας δικτύου. Αυτή η διαδικασία συνεχίζεται μέχρις ότου φθάσουμε στο τελευταίο επίπεδο του δικτύου μας, όπου παρουσιάζεται η εξής διαφορά: η συνάρτηση ενεργοποίησης αυτήν την φορά είναι η softmax, όπου μας δίνει τις πιθανότητες το παράδειγμα που έδωσε αυτό το αποτέλεσμα να ανήκει σε κάθε μία από τις κλάσεις. Η μοναδική διαφορά που παρουσιάζεται στην κλήση αυτής της συνάρτησης στην διαδικασία εκπαίδευσης από την διαδικασία εκτίμησης είναι ότι για την ακρίβεια του μοντέλου τόσο στα δεδομένα εκπαίδευσης όσο και στα δεδομένα εκτίμησης χρησιμοποιούνται όλα τα δείγματα τους και όχι ένα minibatch αυτών. Από την συνάρτηση αυτή επιστρέφονται οι κανονικοποιημένοι έξοδοι των κρυφών νευρώνων (για τον εξωτερικό νευρώνα απλά η έξοδος του μοντέλου), όπως και οι τιμές των εξόδων τους πριν την συνάρτηση ενεργοποίησης του κάθε επιπέδου.

Back propagation: Στην συνέχεια φθάνουμε στο πιο δύσκολο σημείο του κώδικα μας που αφορά την μέση τιμή του διανύσματος κλίσης στο σύνολο του εκάστοτε minibatch ώστε να γίνει η ανανέωση των βαρών για την ελαχιστοποίηση του συνολικού κόστους. Η δυσκολία δεν αναφέρεται στην υλοποίηση του κώδικα (αν και απαιτείται πολύ προσοχή), αλλά κυρίως στην βαθύτατη κατανόηση του μαθηματικού υπόβαθρου και της διαδικασίας που πρέπει να κάνουμε. Στον κώδικα μου λοιπόν υπολογίζω το gradient του κάθε επιπέδου του νευρωνικού ξεχωριστά από έξω προς τα μέσα αυτήν την φορά προτού το πολλαπλασιάσω με την είσοδο που το προκάλεσε για να πάρω στην συνέχεια την μέση τιμή που θα οδηγήσει στην τελική ανανέωση των βαρών. Πιο συγκεκριμένα πριν ξεκινήσουμε την εξήγηση πρέπει πρώτα να μετατρέψουμε την επιθυμητή έξοδο του μοντέλου μας σε one hot encoded. Αφού γίνει αυτό για το δέλτα εξόδου θα χρειαστούμε την παράγωγο της συνάρτησης εξόδου ως προς την έξοδο του μοντέλου μας πολλαπλασιασμένη element-wise με την παράγωγο της συνάρτησης ενεργοποίησης του εξωτερικού επιπέδου προς την είσοδο που αυτή έλαβε (dz). Στην συνέχεια για το δέλτα των βαρών θα υπολογίσω το εσωτερικό γινόμενο του δέλτα εξόδου με την είσοδο του εξωτερικού επιπέδου και θα διαιρέσω με το πλήθος των δειγμάτων του minibatch ώστε να πάρω την μέση τιμή στο σύνολο των δειγμάτων (dw). Αρκετά πιο απλά για το δέλτα του bias απλά θα πάρω την μέση τιμή του dz για κάθε νευρώνα στο σύνολο των δειγμάτων (db). Για τα κρυφά επίπεδα η διαφορά βρίσκεται στο υπολογισμό του δέλτα εξόδου καθώς από εκείνο το σημείο και έπειτα ο υπολογισμός των δέλτα των βαρών και των bias ακολουθεί την ίδια τακτική. Για τον υπολογισμό του δέλτα σε κρυφό επίπεδο χρειαζόμαστε το εσωτερικό γινόμενο του δέλτα του προηγούμενου επιπέδου που υπολογίστηκε (πρακτικά δηλαδή του επόμενου επιπέδου) με τα αντίστοιχα βάρη του επόμενου επιπέδου. Η συνάρτηση αυτή χρησιμοποιείται μόνο κατά την περίοδο εκπαίδευσης και επιστρέφει τα δέλτα των βαρών και των bias ώστε να πάμε στο επόμενο και τελευταίο στάδιο της εκπαίδευσης, την ανανέωση των βαρών.

Update params: Στο τελικό στάδιο της εκπαίδευσης η ανανέωση βαρών και των bias γίνεται με τον εξής τρόπο:

wnew = wold – lr(dw + (1e-x) \* wπαλιο)

bnew = bold – lr(db + (1e-x) \* bπαλιο)

όπου lr = βήμα εκπαίδευσης, w = τα βάρη των νευρώνων ,b = τα bias των νευρώνων και τα x που δοκιμάστηκαν είναι 2 και 3 (δοκιμάστηκε ο κώδικας και χωρίς αυτά αλλά προτιμήθηκε αυτή η υλοποίηση για λόγους που θα εκφραστούν στην συνέχεια)

Για να εξηγήσω καλύτερα κάποιες πράξεις ο λόγος που χρησιμοποιείται το μείον, όπως εξηγείται και από την θεωρεία, οι κλίσεις δέλτα δείχνουν προς την αύξηση της συνάρτησης κόστους, ενώ εμείς θέλουμε να πάμε προς την κατεύθυνση μείωσης αυτής. Επιπλέον ο όρος (1e-x) \* wπαλιο χρησιμοποιείται για να γίνει L2 κανονικοποίση των βαρών αφαιρώντας ένα ποσοστό των βαρών ώστε το μοντέλο να αποφύγει κατά το δυνατόν φαινόμενα overfitting. Στο τελικό μου μοντέλο χρησιμοποιήθηκε αυτός ο όρος με x = 2, ενώ θα παραθέσω και διαγράμματα που δεν χρησιμοποιήθηκε για να ελέγξουμε την διαφορά.

Στο τελευταίο στάδιο του κώδικα εκτυπώνω ανά συχνές περιόδους (ανά δέκα εποχές) αποτελέσματα όπως το κόστος του μοντέλου μου και την ακρίβεια τόσο στα δεδομένα εκπαίδευσης, όσο και στα δεδομένα εκτίμησης. Στο τέλος δημιουργώ δύο διαγράμματα για να δω πως κινήθηκαν τα κόστη και οι 2 ακρίβειες κατά την πάροδο των εποχών.

Σημείωση: στον κώδικα που παραδίδω το έχω βάλει μια συνάρτηση όπου ανά συγκεκριμένες εποχές μου μειώνει το βήμα εκπαίδευσης ώστε να κινηθώ να γίνει αρχικά γρήγορη σύγκλιση και στην πορεία πιο σταθερή ώστε συνέχεια να μειώνω το κόστος και να μην υπάρξει περίπτωση αυξομείωσης του.

Παρατηρήσεις

Κατά την ενασχόληση μου με των κώδικα διαπίστωσα κάποιες δυσκολίες. Πολλές φορές οι τιμές που έπαιρνα στις εξόδους μου ή ήταν πολύ μεγάλες ώστε να μετατρέπονται από την python σε nan η αντίστοιχα πολύ μικρές ώστε να τις μηδενίζει, πράγματα που με το που γινόντουσαν κατέστρεφαν το πρόγραμμά μου καθώς ο κώδικας δεν λειτουργούσε πλέον σωστά. Για τον λόγο αυτό κρίθηκε αναγκαία η κανονικοποίηση των εισόδων του κάθε επιπέδου όπως αναφέρθηκε πάνω, καθώς και αν κάποιες τυπικές αποκλίσεις ήταν μικρότερες από κάποιο κατώφλι (αρκετά πιθανό σενάριο, που μου συνέβη) να γίνονται ίσες με το κατώφλι ώστε να μπορεί να συνεχίσει χωρίς πρόβλημα το πρόγραμμα μου. Άλλες φορές παρουσιάστηκαν προβλήματα overfitting που αντιμετωπίστηκαν με τον τρόπο που αναφέρθηκε παραπάνω. Καθώς και προβλήματα όπως η αυξομείωση της συνάρτησης κόστους σε περιπτώσεις όπου δοκιμάστηκαν μεγάλα βήματα εκπαίδευσης. Γεγονός που με οδήγησε στην μείωση το βήματος μάθησης ανά κάποιες εποχές.

Συμπεράσματα

Για αρχή θεωρώ σημαντικό να αναφέρω ότι καθώς υλοποιώ κώδικα MLP from

scratch ο κώδικας δεν θα μπορούσε να χρησιμοποιηθεί, παρά μόνο για εκπαιδευτικούς λόγους, για καλύτερη κατανόηση της λειτουργίας των μοντέλων αυτών. Είναι αρκετά αργός κώδικας, χρειάζεται ώρες να τρέξει, αρκετά μεγαλύτερο χρόνο σε σχέση με αντίστοιχους κώδικες που υλοποίησαν συμφοιτητές μου χρησιμοποιώντας frameworks που τρέχουν το πρόγραμμα στην gpu και με βελτιστοποιημένες συναρτήσεις.

Μέσω τον αρκετών παραδειγμάτων που δοκίμασα συμπέρανα επίσης ότι η κάθε βάση δεδομένων έχει τα δικά της μοναδικά χαρακτηριστικά, πράγμα που σημαίνει ότι δεν έχει σημασία η ποσότητα των νευρώνων που υλοποιείς. Άλλα dataset λειτουργούν καλύτερα με πολλά κρυφά επίπεδα και με πολλούς νευρώνες το καθένα, ενώ κάποια άλλα και με λίγους νευρώνες και με 1-2 κρυφά επίπεδα, άλλα με μεγάλα batch size, άλλα με μικρότερα. Συγκεκριμένα, αν και δοκίμασα και τρίτο κρυφό επίπεδο ή ακόμα και με 2 κρυφά επίπεδα με περισσότερους νευρώνες στο 2ο κρυφό επίπεδο, συμπέρανα ότι η καλύτερη δομή για το πρόβλημα μου προήλθε από 3072 νευρώνες εισόδου, 50 νευρώνες στο κρυφό επίπεδο και 10 νευρώνες στο επίπεδο εξόδου και με batch size = 250.

Παραδείγματα

1ο παράδειγμα με 3072 νευρώνες εισόδου, 50 κρυφούς νευρώνες και 10 νευρώνες εξόδου χωρίς την L2 κανονικοποίηση και με batch-size =250. Φαίνεται ξεκάθαρα το overfitting που συμβαίνει καθώς η διαφορά των 2 accuracy «ακουμπά» το 20%

Εικόνα που περιέχει κείμενο, γράφημα, γραμμή, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα

Στο επόμενο παράδειγμα με κάπως μεγαλύτερο batch size = 400 και πάλι χωρίς L2 κανονικοποίηση βλέπουμε αρχικά πόσο μεγαλύτερη γίνεται η διαφορά των δύο accuracy ενώ και με το συγκεκριμένο learing rate ότι υπάρχουν αυξομειώσεις της συνάρτησεις κόστους γεγονός που απαιτεί μείωση του. Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γράφημα, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, γράφημα, διάγραμμα, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Στα επόμενα δύο διαγράμματα που παραθέτω έχω ένα νευρωνικό δίκτυο που λειτουργεί με τρία κρυφά επίπεδα πριν το επίπεδο εξόδου με 3072, 200, 50, 10 νευρώνες αντίστοιχα στο όποιο κάνω και L2 κανονικοποιήση (μικρές διαφοροποιήσεις στα δύο accuracies) καθώς επίσης μεταβάλλω και το learning ανά 100 εποχές όπως φαίνεται και στα διαγράμματα. Παρατηρώ επίσης ότι μετά τις 150 εποχές δεν παρατηρούνται σημαντικές διαφορές ούτε στο κόστος ότι στις ακρίβειες του μοντέλου. Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Στα επόμενα διαγράμματα που παραθέτω έχω 2 δύο κρυφά επίπεδα συν το επίπεδο εξόδου με 3072, 50, 10 νευρώνες το καθένα, με L2 κανονικοποίηση και μεταβολή του learning rate όπως και πριν. Παρατηρούμε από την μια πλευρά κάπως μεγαλύτερη διαφορά μεταξύ των δύο accuracy (γύρω στο 4-5% που την θεωρώ επιτρεπτεί), ενώ επιπλέον φθάνουμε χαμηλότερα επίπεδα κόστους και αρκετά υψηλότερα επίπεδα ακρίβειαςΕικόνα που περιέχει κείμενο, γραμμή, στιγμιότυπο οθόνης, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Τέλος να παραθέσω άλλο ένα παράδειγμα ίδιο με προηγούμενο με μοναδική διαφορά ότι το batch size = 400Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Τελικό αποτέλεσμα

Στο τελικό μου αποτέλεσμα παρουσιάζω ένα δίκτυο με 2 κρυφά επίπεδα με 3072 και 50 νευρώνες αντίστοιχα συν το επίπεδο εξόδου με 10 νευρώνες, 250 batch size και τρέχει για 200 εποχές. Η διαφορά που παρουσιάζει με το προ τελευταίο μοντέλο της προηγούμενης ενότητας είναι ότι η L2 κανονικοποίση γίνεται με x = 2 αντί για x = 3 που χρησιμοποιήθηκε σε ολόκληρη την προηγούμενη ενότητα. Μια σημαντική παρατήρηση που μπορούμε να κάνουμε εδώ είναι, αν και ο perceptron μας δουλεύει όπως θα θέλαμε και ελαττώνει συνεχώς το αποτέλεσμα της συνάρτησης κόστους η ακρίβεια των δεδομένων εκτίμησης φαίνεται να φθάνει σε κάποιο κατώφλι. Παρουσιάζεται φαινόμενο overfitting καθώς αν και αυξάνεται συνεχώς με μικρά βήματα η ακρίβεια των δεδομένων εκπαίδευσης, η ακρίβεια των δεδομένων εκτίμησης είναι σταθερή. Ίσως μετά από μεγάλο αριθμών εποχών καταφέρουμε να πετύχουμε πολλή υψηλή ακρίβεια, καθώς τα δεδομένα εκτίμησης, σύμφωνα με την ιστοσελίδα που δόθηκε για τον σκοπό της εργασίας, αποτελούν ένα τυχαίο δείγμα του 20% του συνόλου εκπαίδευσης γεγονός, όμως που θα σήμαινε κατά πάσα πιθανότητα ότι έχει γίνει μεγάλο overfitting στα δεδομένα και ότι το μοντέλο δεν θα είχε ευρεία χρήση. Σε μια γενική περίπτωση ίσως θα σταματούσαμε τον κώδικα μας κοντά στις 150 εποχές όπου η συνάρτηση κόστους αρχίζει να γίνεται οριζόντια.

Epoch 10 / 200

Train Error: 1.481071691688803

Train Accuracy: 0.50942

Test Accuracy: 0.4594

Epoch 20 / 200

Train Error: 1.444894899304525

Train Accuracy: 0.51484

Test Accuracy: 0.4853

Epoch 30 / 200

Train Error: 1.373369428581694

Train Accuracy: 0.53498

Test Accuracy: 0.5021

Epoch 40 / 200

Train Error: 1.280016752936163

Train Accuracy: 0.57174

Test Accuracy: 0.5193

Epoch 50 / 200

Train Error: 1.1999632425180538

Train Accuracy: 0.60946

Test Accuracy: 0.5352

Epoch 60 / 200

Train Error: 1.1128033242450357

Train Accuracy: 0.64662

Test Accuracy: 0.5404

Epoch 70 / 200

Train Error: 1.0460039835346087

Train Accuracy: 0.678

Test Accuracy: 0.5419

Epoch 80 / 200

Train Error: 0.9985221258593537

Train Accuracy: 0.7013

Test Accuracy: 0.5444

Epoch 90 / 200

Train Error: 0.9669212246843764

Train Accuracy: 0.71668

Test Accuracy: 0.5401

Epoch 100 / 200

Train Error: 0.9080032090251449

Train Accuracy: 0.7497

Test Accuracy: 0.5475

Epoch 110 / 200

Train Error: 0.8857472800167799

Train Accuracy: 0.76258

Test Accuracy: 0.5461

Epoch 120 / 200

Train Error: 0.876327386614045

Train Accuracy: 0.76856

Test Accuracy: 0.5452

Epoch 130 / 200

Train Error: 0.868800645913043

Train Accuracy: 0.77362

Test Accuracy: 0.5454

Epoch 140 / 200

Train Error: 0.8627568380471908

Train Accuracy: 0.77766

Test Accuracy: 0.5433

Epoch 150 / 200

Train Error: 0.8482658549938974

Train Accuracy: 0.78536

Test Accuracy: 0.5425

Epoch 160 / 200

Train Error: 0.846015710020003

Train Accuracy: 0.78616

Test Accuracy: 0.5431

Epoch 170 / 200

Train Error: 0.8448904955724571

Train Accuracy: 0.78682

Test Accuracy: 0.543

Epoch 180 / 200

Train Error: 0.8438657593121675

Train Accuracy: 0.78722

Test Accuracy: 0.5431

Epoch 190 / 200

Train Error: 0.8429215868352464

Train Accuracy: 0.78792

Test Accuracy: 0.5427

Epoch 200 / 200

Train Error: 0.8416694555160962

Train Accuracy: 0.78876

Test Accuracy: 0.5428

Εικόνα που περιέχει κείμενο, γραμμή, γράφημα, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει κείμενο, γραμμή, γράφημα, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα