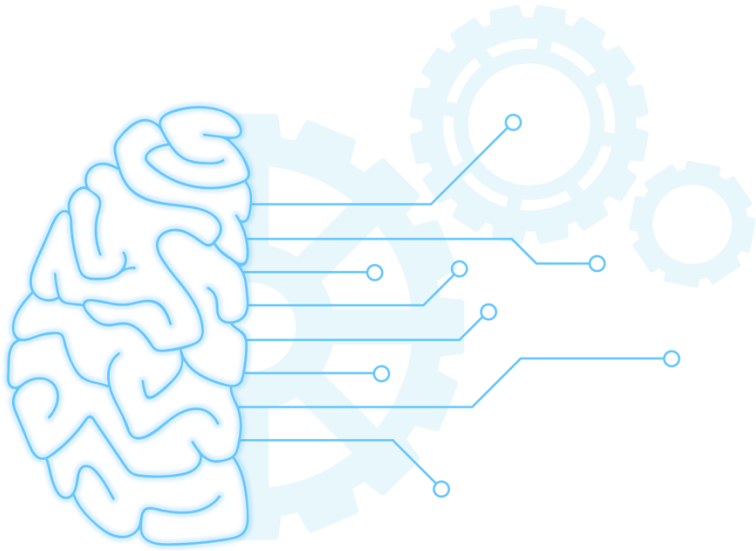
ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ. ΚΑΙ ΜΗΧ. ΥΠΟΛΟΓΙΣΤΩΝ

ΣΤΑΤΙΣΤΙΚΗ ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΚΑΙ ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ

ΤΗΛ 311



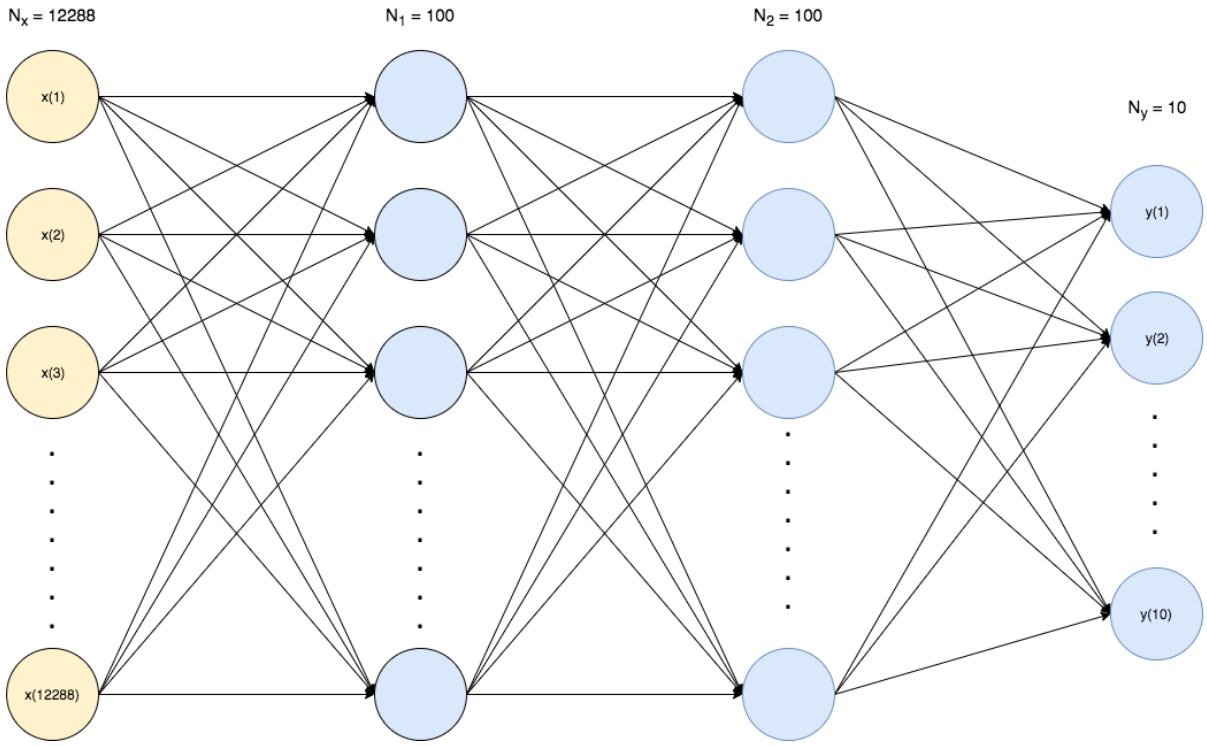
2η ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ
ΑΝΑΦΟΡΑ

ΚΥΡΙΑΖΑΚΗΣ ΚΛΕΑΝΘΗΣ – 2015030086

ΘΕΜΑ 1 | Αρχιτεκτονική Νευρωνικών Δικτύων

Στην άσκηση αυτή θα εξετάσουμε την αρχιτεκτονική των νευρωνικών δικτύων. Πιο συγκεκριμένα, ως είσοδο έχουμε RGB εικόνες διάστασης 64x64 οπότε το πλήθος των στοιχείων εισόδου του νευρωνικού δικτύου μας είναι: 3x64x64 = 12288.

Στο πρώτο και δεύτερο hidden layer έχουμε 100 κόμβους έκαστος και ως έξοδο έχουμε 10 κόμβους που ο καθένας αντιστοιχεί σε κάθε κλάση. Το νευρωνικό δίκτυο φαίνεται παρακάτω:



Το κάθε στοιχείο εισόδου συνδέεται με τον κάθε ένα από τους κόμβους του πρώτου hidden layer με ένα μοναδικό βάρος (το x(1) συνδέεται με τους 100 κόμβους του πρώτου hidden layer με 100 ξεχωριστά βάρη).

Οπότε στο σημείο μεταξύ εισόδου και 1ου hidden layer έχουμε:

12288x100 = 1.228.800 ξεχωριστά βάρη.

Όμοια, στο σημείο μεταξύ 1ου και 2ου hidden layer έχουμε:

100x100 = 10.000 ξεχωριστά βάρη.

Τέλος, στο σημείο μεταξύ 2ου hidden layer και εξόδου έχουμε:

100x10 = 1.000 ξεχωριστά βάρη.

Οπότε συνολικά, ο αριθμός των παραμέτρων είναι:

$$1.228.800 + 10.000 + 1.000 = 1.239.800$$

ΘΕΜΑ 2 | Λογιστική Παλινδρόμηση. Αναλυτική εύρεση κλίσης (Gradient)

α) Στο πρώτο μέρος του 2ου θέματος, έχουμε ένα σύνολο m δεδομένων x_i καθώς και τα label τους y_i. Επίσης, μας δίνεται η συνάρτηση λογιστικής παλινδρόμησης $h_\theta(x) = f(\theta^T x)$ όπου f(z) η λογιστική σιγμοειδής συνάρτηση $f(z) = \frac{1}{1+e^{-z}}$

Έχοντας γνωστό τον τύπο της συνάρτησης κόστους (loss function), ζητείται ο υπολογισμός της κλίσης gradient του σφάλματος J(θ).

Λύση:

Αρχικά υπολογίζουμε τον παρακάτω λογάριθμο ο οποίος θα μας χρειαστεί αργότερα:

$$\ln\left[\mathrm{h}_\theta\left(x^i\right)\right] = \ln\left(\frac{1}{1+e^{-\theta^T \cdot x^i}}\right) = -\ln\left(1+e^{-\theta^T \cdot x^i}\right) \quad \Sigma\chi\acute{\epsilon}\sigma\eta\ (1)$$

$$\text{Οπότε έχουμε:} \quad J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^i \cdot \ln\left(\mathrm{h}_\theta\left(x^i\right)\right) + \left(1-y^i\right) \cdot \ln\left(1-\ \mathrm{h}_\theta\left(x^i\right)\right)\right]$$

Πριν γίνει η παραγώγιση θα πρέπει να το απλοποιήσουμε:

$$\overset{(1)}{\Rightarrow} J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[-y^i \cdot \ln\left(1 + e^{-\theta^T \cdot x^i}\right) + \left(1 - y^i\right) \cdot \left(-\theta^T \cdot x^i - \ln\left(1 + e^{-\theta^T \cdot x^i}\right)\right)\right]$$

Μετά από παραγοντοποίηση:

$$= -\frac{1}{m}\sum_{i=1}^{m}\left[y^i \theta^T x^i - \theta^T x^i - \ln\left(1 + e^{-\theta^T \cdot x^i}\right)\right]$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left[y^i \theta^T x^i - \ln\left(1 + e^{\theta^T \cdot x^i}\right)\right]$$

$$\Rightarrow J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^i \theta^T x^i - \ln\left(1 + e^{\theta^T \cdot x^i}\right)\right]$$

Άρα τώρα παραγωγίζουμε δύο μέρη:

$$\frac{\partial}{\partial \theta_J}[y^i \theta^T x^i] \quad και \quad \frac{\partial}{\partial \theta_J}\left[\ln\left(1 + e^{\theta^T \cdot x^i}\right)\right]$$

$$Πρώτα: \boxed{\frac{\partial}{\partial \theta_J}[y^i \theta^T x^i] = y^i x_j^i} \; Σχέση\ (2)$$

$$Έπειτα: \frac{\partial}{\partial \theta_J}\left[\ln\left(1 + e^{\theta^T \cdot x^i}\right)\right] = \frac{x_j^i \cdot e^{\theta^T \cdot x^i}}{1 + e^{\theta^T \cdot x^i}} = \frac{x_j^i}{e^{\theta^T \cdot x^i} \cdot \left(1 + e^{\theta^T \cdot x^i}\right)}$$

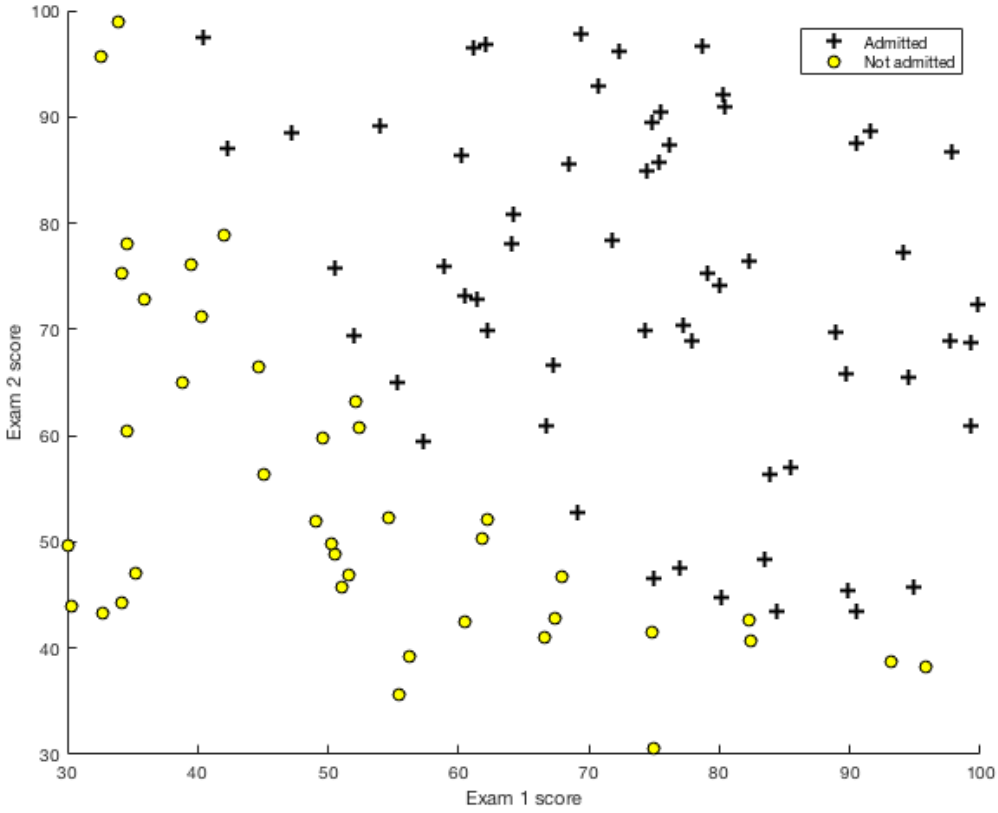
$$= \frac{x_j^i}{e^{\theta^T \cdot x^i} + e^{-\theta^T \cdot x^i + \theta^T \cdot x^i}} = x_j^i \cdot \frac{1}{e^{\theta^T \cdot x^i} + 1} = x_j^i \cdot \mathrm{h}_\theta\left(x^i\right)$$

$$\boxed{\frac{\partial}{\partial \theta_J}\left[\ln\left(1 + e^{\theta^T \cdot x^i}\right)\right] = x_j^i \cdot \mathrm{h}_\theta\left(x^i\right)} \; Σχέση\ (3)$$

$$Από\ (2), (3): \quad \frac{\partial J(\theta)}{\partial \theta_J} = \frac{1}{m}\sum_{i=1}^{m}\left(\mathrm{h}_\theta\left(x^i\right) - y^i\right) \cdot x_j^i$$

b) Στο δεύτερο μέρος χρησιμοποιούμε τα δεδομένα από το αρχείο exam_scores_data.txt το οποίο περιέχει πληροφορία για τους βαθμούς σε δύο εξετάσεις για 100 φοιτητές καθώς και το αν γίνανε δεκτοί σε ένα πανεπιστήμιο.Έτσι, θα κάνουμε train ένα μοντέλο το οποίο με βάση τους δύο βαθμούς θα προβλέπει το αν μελλοντικοί φοιτητές θα γίνονται δεκτοί στο πανεπιστήμιο.

Αρχικά, στο Part 1 του κώδικα γίνεται η ανάγνωση του αρχείου και η εμφάνιση των δεδομένων σε κοινό plot έτσι ώστε να διακρίνουμε οπτικά τα δεδομένα μας.



Από το παραπάνω διάγραμμα παρατηρούμε πως θα μπορούσε να υπάρξει ένας γραμμικός τρόπος να διακρίνουμε τις περιπτώσεις (αποδοχή/απόρριψη) που σημαίνει ότι τα feature που έχουμε είναι καλά.

Στο Part 2 του κώδικα γίνεται η κλήση της συνάρτησης 'costFunction' την οποία κληθήκαμε να υλοποιήσουμε με παραμέτρους τα δείγματα X, τα label τους Y καθώς και του θ (που ορίσαμε ως πίνακα 1x3 με μηδενικά).

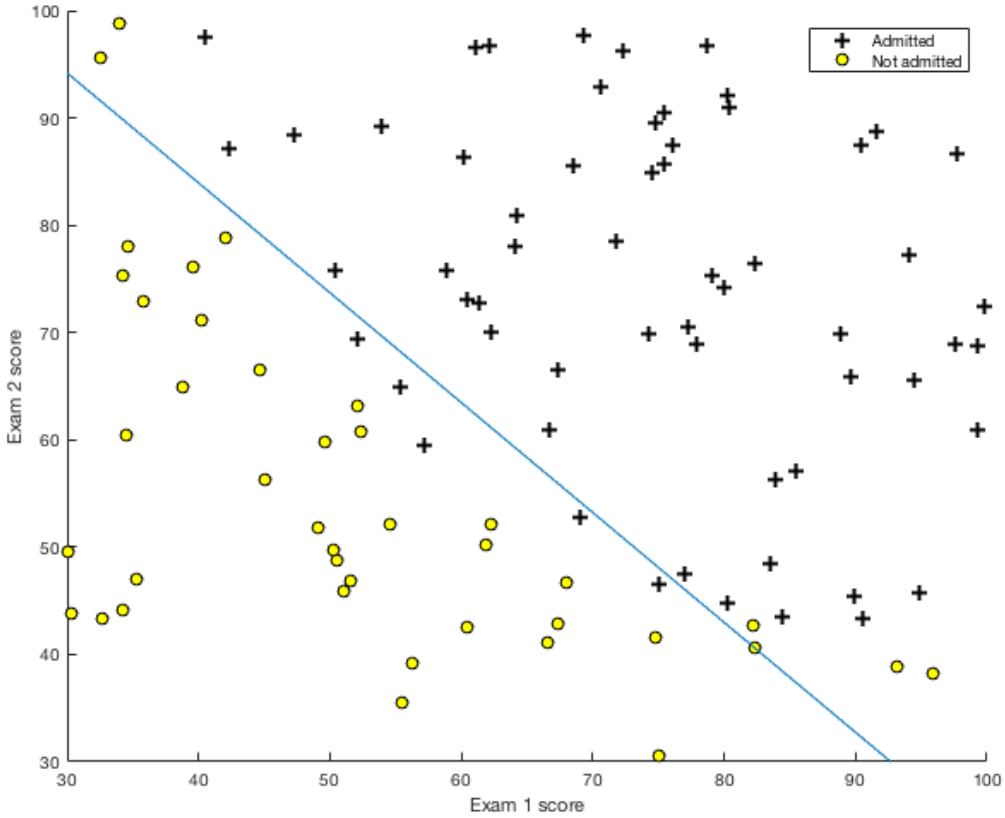
Πριν γίνει η υλοποίηση της 'costFunction' όμως έπρεπε να ορίσουμε την σιγμοειδή συνάρτηση 'sigmoid' η οποία υλοποιεί τον τύπο f(z) που μας δόθηκε στο α ερώτημα. Τώρα πρέπει να προσαρμόσουμε την συνάρτηση κόστους ώστε να υπολογίζει και να επιστρέφει το κόστος καθώς και το gradient. Αρχικά, καλούμε την 'sigmoid' με όρισμα το X*Theta έτσι ώστε να χρησιμοποιηθεί ως $\hat{y}^{(i)}$ στην συνάρτηση J(θ) για τον υπολογισμό του κόστους καθώς και της κλίσης με τους τύπους:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(-y^{(i)} \cdot \ln\left(\hat{y}^{(i)}\right) - \left(1 - y^{(i)}\right) \cdot \ln\left(1 - \hat{y}^{(i)}\right)\right)$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)x_j^{(i)}$$

Έτσι, υπολογίζεται και επιστρέφεται το κόστος των συγκεκριμένων δεδομένων J = 0.693147 και η κλίση grad = [-0.1, -12, -11.26] που συμπίπτουν με τα νούμερα που μας δόθηκαν στην εκφώνηση.

Στο Part 3 του κώδικα γίνεται ο υπολογισμός του βέλτιστου Theta και ξανά υπολογίζεται το κόστος με την συνάρτηση που φτιάξαμε πριν. Το κόστος πλέον με την χρήση theta = [-24.93, 0.20, 0.19] είναι J = 0.203506. Τέλος βρέθηκε το σύνορο απόφασης και εμφανίζεται παρακάτω.



Τέλος, στο Part 4 δοκιμάζουμε τον αλγόριθμο με έναν μαθητή που έχει γράψει 45 στην πρώτη εξέταση και 85 στην δεύτερη, για να προβλέψουμε την πιθανότητα αποδοχής του στο πανεπιστήμιο.

Το αποτέλεσμα που παίρνουμε είναι ότι η πιθανότητα αποδοχής είναι 77.4321% καθώς επίσης και το train accuracy 89%.

ΘΕΜΑ 3 | Νευρώνες Sigmoid και SoftMax: Βελτιστοποίηση κατά προσέγγιση χρησιμοποιώντας SGD

Σε αυτό το θέμα χρησιμοποιούμε το ίδιο dataset με το προηγούμενο ερώτημα για να προβλέψουμε αν ένας φοιτητής θα γίνει δεκτός στο πανεπιστήμιο, χρησιμοποιώντας όμως επαναληπτικά τον SGD αλγόριθμο. Για την υλοποίηση αυτού του προβλήματος χρησιμοποιήθηκε η 3.7.3 έκδοση της Python και η 1.13.1 έκδοση του TensorFlow. Παρακάτω αναλύονται κάποια κομμάτια κώδικα:

A) Στην πρώτη περίπτωση έχουμε έναν νευρώνα με συνάρτηση ενεργοποίησης την λογιστική συνάρτηση όπως και πριν. Συμπληρώθηκε ο κώδικας στο αρχείο 'mylogLR.py' ως εξής. Στην συνάρτηση 'normalize' η μέση τιμή και η τυπική απόκλιση υπολογίστηκαν με τις numpy.mean() και numpy.std() αντίστοιχα, η κανονικοποίηση έγινε με τον τύπο $\frac{x-\bar{x}}{\sigma}$.

Ο ορισμός του γραμμικού μοντέλου γίνεται από τον τύπο X*W + B οπού ο πολλαπλασιασμός γίνεται με την συνάρτηση matmul() του Tensorflow.

Το classification γίνεται με την χρήση της tf.sigmoid() για τον υπολογισμό της πιθανότητας και έπειτα γίνεται η σύγκριση αυτής της πιθανότητας με το threshold για να γίνει η λήψη της απόφασης.

a. Η δημιουργία του μοντέλου γίνεται μέσα στο tf.Session() με 55 epochs, υπολογίζοντάς σε κάθε επανάληψη το epoch_loss. Μετά το Train, γίνεται αξιολόγηση του μοντέλου με τα δεδομένα [45, 85]. Το αποτέλεσμα είναι 1:αποδοχή με πιθανότητα 68.3% που προσεγγίζει την τιμή που πήραμε στο προηγούμενο ερώτημα.

b. Στη συνέχεια καλέσαμε την συνάρτηση evaluate( ) με τα δεδομένα εκπαίδευσης [X_np, Y_np] και τρέξαμε το πείραμα πολλές φορές για να παρατηρήσουμε την απόδοση. Πήραμε ένα εύρος train accuracy 85 με 90% που συμπίπτει με το train accuracy του Θέματος 2.

c. Στη συνέχεια έγινε δοκιμή του πειράματος με διάφορες τιμές threshold στο διάστημα [0,1] παρατηρώντας την επίδοση του συστήματος. Για τιμές threshold μικρότερες του 0,2 ή μεγαλύτερες του 0,8 ,η απόδοση έπεφτε σημαντικά στις περιοχές 50-65%.

Με τιμές threshold στο διάστημα [0.4, 0.6] έχουμε καλές τιμές απόδοσης στο εύρος 80-93% οπότε μπορούμε να συμπεράνουμε ότι η βέλτιστη απόφαση για την τιμή του κατωφλίου θα είναι να επιλέξουμε ένα νούμερο κοντά στο 0.5

d. Δοκιμάσαμε την εκπαίδευση με batches μεγέθους 10, 20 και 50 και καταγράψαμε τα εξής αποτελέσματα:

- **Batch size = 10**: Κάθε φορά, η τιμή του epoch_loss ξεκινάει από μία σχετικά υψηλή τιμή (κοντά στο 0.9-1.2) και στις τελευταίες επαναλήψεις έχει πέσει στις τιμές 0.3-0.4

- **Batch size = 20**: Σε αυτή την περίπτωση, το epoch_loss ξεκινάει από μία αρκετά μικρότερη τιμή σε σχέση με πριν (κοντά στο 0.4-0.5) και καταλήγει στις τιμές 0.25-0.35. Παρόλο που δεν έχει σημαντική μείωση σε σχέση με την αρχική του τιμή, βλέπουμε ότι οι τελικές τιμές είναι παρόμοιες αυτές για Batch size = 10.

- **Batch size = 50**: Τέλος, χρησιμοποιούμε τα μισά δείγματά μας για την εκπαίδευση του μοντέλου. Παρατηρούμε πως η τιμή του epoch_loss ξεκινάει από τιμές σημαντικά μικρότερες των προηγούμενων που ανήκουν στο εύρος 0.02-0.04. Μέχρι το τέλος όμως παραμένουν σχεδόν ίδιες με διαφορά $3^{ου}$ ή $4^{ου}$ δεκαδικού ψηφίου.

Όσον αφορά το learning rate, παρατηρούμε πως όσο αυξάνεται τόσο βελτιώνεται η τιμή του epoch_loss. Αυτό όμως μέχρι ένα σημείο (>5), που ξεκινάμε να χάνουμε τα τοπικά ελάχιστα και έχουμε πολύ μεγάλη απώλεια.

Β) Στο δεύτερο ερώτημα, υλοποιήθηκε ο κώδικας όπου αντικαθιστούμε την λογιστική συνάρτηση με την SoftMax. Πλέον χρησιμοποιούμε την συνάρτηση tf.nn.softmax() για τον υπολογισμό των πιθανοτήτων και όχι την sigmoid, καθώς και άλλες συναρτήσεις που μας δίνονται για τον υπολογισμό άλλων παραμέτρων όπως sparse_softmax_cross_entropy_with_logits() ή argmax().

Το μοντέλο που εκπαιδεύουμε εδώ, προβλέπει την αποδοχή του μαθητή με [45, 85] με πιθανότητες [0.44724312 0.55275685] όπως προέβλεψε και το προηγούμενο. Επίσης, το εύρος του train accuracy είναι 86 με 90% που συμπίπτει με τα προηγούμενα. Η μόνη διαφορά είναι ότι η τιμή του epoch_loss ξεκινάει από χαμηλότερη τιμή (0.5-0.7) και στις τελευταίες επαναλήψεις έχει πέσει στις τιμές 0.2-0.3.

Η καλύτερη επιλογή θα ήταν η χρήση της λογιστικής συνάρτησης μιας και το πρόβλημα που έχουμε να λύσουμε είναι δυαδικό (αποδοχή ή απόρριψη). Το SoftMax Regression είναι γενίκευση του Logistic Regression και θα ήταν πιο κατάλληλο σε προβλήματα με περισσότερες κλάσεις στην έξοδο.

ΘΕΜΑ 4 | Feature Selection – Classification – Cross Validation - Overfitting

Στο πρόβλημα αυτό, έχουμε δεδομένα εγκεφαλογραφήματος 25 ατόμων οπού 15 από αυτούς είναι ασθενείς. Για κάθε καταγραφή έχουμε 1000 χαρακτηριστικά που είναι μεγάλος αριθμός σε σχέση με τον μικρό αριθμό δειγμάτων. Χρησιμοποιούμε την τεχνική leave-one-out διότι έχουμε μόνο 25 δείγματα και στο τέλος παίρνουμε την μέση τιμή των αποτελεσμάτων. Αρχικά, χρησιμοποιούνται και τα 1000 χαρακτηρίστηκα και με την παραπάνω τεχνική μετράμε την επίδοση του συστήματος.

Στο Classification χωρίς feature selection έχουμε επίδοση 52% και για αυτό αποφασίζουμε να κάνουμε feature selection έτσι ώστε να επιλέξουμε τα 100 χαρακτηριστικά που διαχωρίζουν περισσότερο τους ασθενείς με του υγιής. Χρησιμοποιούμε τον συντελεστή συσχέτισης Pearson που μας δίνεται στην εκφώνηση και ορίζεται στο αρχείο ‘similarityMeasure.m’.

Για το ερώτημα C κάνουμε feature selection μέσα στο cross validation, έπειτα κάνουμε ταξινόμηση του πίνακα συσχετίσεων και κρατάμε τα πρώτα 100 χαρακτηριστικά. Θεωρητικά τώρα βασισμένοι στα 100 καλύτερα χαρακτηριστικά θα πρέπει να βελτιωθεί η απόδοση του συστήματός μας, και όντως πλέον η απόδοση είναι 72%.

Τέλος, στο ερώτημα D, γίνεται feature selection πριν το cross validation για όλα τα 25 δείγματα που έχουμε. Έτσι, παρουσιάζεται overfit διότι λαμβάνουμε υπόψιν και το 25ο δείγμα στην επιλογή χαρακτηριστικών. Άρα περιμένουμε βελτίωση στην απόδοση του συστήματος η οποία πλέον είναι 100%.

Γενικά, η τακτική leave-one-out χρησιμοποιείται για να μην παρουσιάζονται προβλήματα overfitting όταν έχουμε μικρό αριθμό δειγμάτων. Στην τελευταία περίπτωση αν δεν χρησιμοποιούσαμε το 25ο δείγμα που αφήναμε κάθε φορά εκτός δεν θα υπήρχε overfitting και έτσι δεν θα είχαμε 100% απόδοση. Επίσης, οι τιμές των αποδόσεων δεν είναι αντιπροσωπευτικές διότι με κάθε εκτέλεση υπάρχουν εντελώς διαφορετικά αποτελέσματα πράγμα που οφείλεται στον τρόπο που κάνουμε generate τα δεδομένα μας (randn).