ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ ΙΙ

ΕΡΓΑΣΙΑ 2

Πρόλογος

Σκοπός της εργασίας αυτής είναι να προπονηθεί ένα νευρωνικό δίκτυο και συγκεκριμένα ένα **feed-forward** για ένα πρόβλημα κατηγοριοποίησης πολλαπλών κλάσεων με σκοπό την περαιτέρω ανάλυση του χαρακτήρα (**sentiment analysis**) κάποιων αναρτήσεων του twitter. Στην συνέχεια, θα αναλυθούν λεπτομερώς τόσο τα βήματα που ακολουθήθηκαν όσο και κάποιες παρατηρήσεις αλλά και συγκρίσεις μεταξύ διαφορετικών πρακτικών.

Γενική περιγραφή

Τα πρώτα βήματα που ακολουθήθηκαν είναι αρκετά παρόμοια με αυτά της προηγούμενης εργασίας δηλαδή η **φόρτωση** των training και validation **datasets**, η **προεπεξεργασία** τους και η εξαγωγή **στατιστικών στοιχείων** είναι σχεδόν τα ίδια. Η μόνη διαφορά είναι πως η προεπεξεργασία διαθέτει τώρα δύο επιλογές όπου η μια είναι η επιλογή της **απλής** επεξεργασίας και η άλλη της **πιο λεπτομερούς** επεξεργασίας (αυτή κυρίως θα χρησιμοποιηθεί) για περισσότερο καθάρισμα π.χ stopwords, μικρές λέξεις κτλ. Στην συνέχεια, υπάρχουν κάποιες συναρτήσεις που αφορούν τα **γραφήματα** τα οποία θα παραχθούν παρακάτω για την γραφική απεικόνιση κάποιων **μετρικών απόδοσης** (confusion matrices, learning curves/loss vs epochs graph, roc curves).

Ύστερα ακολουθούν μερικές δηλώσεις σχετικά με κάποια αναγκαία δομικά στοιχεία του νευρωνικού.

Πιο συγκεκριμένα, παρουσιάζονται δύο αρκετά συνηθισμένες με καλές ιδιότητες **activation functions** της ίδιας οικογένειας, η **ReLU** και η **SELU,** που πρόκειται να χρησιμοποιηθούν στα ενδιάμεσα επίπεδα (**hidden layers**).

Επίσης, δηλώνονται και δύο **loss functions** όπου η μια είναι η **Cross Entropy loss** που είναι η κατά κόρον πιο χρησιμοποιούμενη, για τις καλές της ιδιότητες, σε προβλήματα κατηγοριοποίησης πολλαπλών κλάσεων και επιπλέον θα δοκιμαστεί για τις ιδιότητες της και η **Hinge Multinomial loss (Multi Margin Loss)** η οποία εντάσσεται στην κατηγορία των losses με margin και ουσιαστικά είναι μια επέκταση της **maximum objective function** για κατηγοριοποίηση σε πολλαπλές κλάσεις (παρουσιάστηκε και στο μάθημα σε ένα **NLP** παράδειγμα μεταξύ δύο κλάσεων ώστε

να μεγιστοποιηθεί η διαφορά των προβλέψεων/scores της πραγματικής κλάσης από την άλλη + κάποιο περιθώριο (margin)) .

Επιπλέον, υπάρχουν οι δηλώσεις κάποιων αρκετά γνωστών **optimizers** οι οποίοι παραμετροποιούνται ως προς τις **παραμέτρους** του **μοντέλου** που θα εφαρμοστούν, ως προς το **learning rate** και ως προς τον βαθμό **regularization** των βαρών (**weights)** του μοντέλου με **L2 νόρμα**. Χρησιμοποιείται ως γνωστόν ο **Stohastic Gradient Decent** αλλά και κάποιες στενές παραλλαγές του (**Momentum** (χρησιμοποιείται έμμεσα μέσω του One Cycle schelduler), **Nesterov**) για πιο καλή και γρήγορη σύγκλιση λόγω του **exponential decay** των gradients αποφεύγοντας με αυτόν τον τρόπο **τοπικά** ελάχιστα/βέλτιστα που πιθανώς ο πρώτος να συναντήσει. Ταυτόχρονα, υπάρχουν και δύο **adaptive** optimizers , οι οποίοι **προσαρμόζουν το learning rate** από μόνοι τους και ο ένας είναι ο **Adadelta** ο οποίος είναι μια βελτιωμένη έκδοση του **Adagrad** έτσι ώστε να αποφεύγει τις μεγάλες και απότομες αλλαγές του learning rate που οδηγούν σε πολύ γρήγορη σύγκλιση και ο άλλος ο πασίγνωστος **Adam** που είναι πιο πολύ ένας υβριδικός (**hybrid**) (adaptive + exponential decay).

Τέλος, για να βοηθηθούν περισσότερο κυρίως οι μη adaptive optimizers κατασκευάστηκαν και κάποιοι **scheldulers** για την **δυναμική προσαρμογή του learning rate** με αποτέλεσμα τώρα να μπορούν να αποφευχθούν πιο εύκολα τα τοπικά βέλτιστα εφόσον οι τιμές των διαφορετικών learning rates μπορούν να επιλέγονται σε ένα πιο χαλαρό εύρος τιμών ξεκινώντας κυρίως από μεγάλες/μικρές τιμές οι οποίες όμως μικραίνουν/αυξάνονται στην πορεία. Οι scheldulers αυτοί υλοποιήθηκαν με τέτοιον τρόπο όπως παρουσιάστηκαν στο μάθημα και υλοποιήθηκαν οι περισσότεροι όπως ο **Exponential**, ο **Power**, ο **Piecewise**, ο **Performance** και ο **One Cycle** schelduler. Οι scheldulers αυτοί παραμετροποιούνται όλοι ως προς τον optimizer του μοντέλου ώστε του αλλάξουν το learning rate που κατέχει και ως προς κάποιες δικές τους παραμέτρους ώστε να προσαρμόζουν τα άλματα του learning rate.

Συνεχίζοντας ορίζεται η κλάση που θα χρησιμοποιηθεί για την μοντελοποίηση του νευρωνικού και την αυτοματοποίηση λειτουργιών του και διαχωρίζεται στα εξής βασικά σημεία:

- **Αρχικοποίηση** (__init__): Σε αυτό το στάδιο αρχικοποιείται η κλάση και κατασκευάζεται όλη η **δομή του feed-forward** νευρωνικού. Ουσιαστικά, δίνονται στην αρχή τα ενδιάμεσα επίπεδα (**hidden layers**) με την αντίστοιχη σειρά και το τελευταίο επίπεδο (**output layer**) στο τέλος, με κάθε επίπεδο να αναπαρίσταται ως προς τον αριθμό των νευρώνων (**neurons**) του αλλά και την **activation function** που θα εφαρμοστεί στην έξοδο του. Αν στο τελευταίο επίπεδο δεν δίνεται (π.χ όταν θα χρησιμοποιηθεί ως loss function η Cross Entropy) για να μπορέσει το μοντέλο να προβλέπει τιμές αν

χρειαστεί δηλώνεται ένα **Softmax** layer. Οπότε δίνοντας και το μέγεθος της εισόδου γίνεται η σύνδεση αυτόματα με την σειρά που δόθηκαν τα επίπεδα (με τη χρήση του ModuleList ώστε να είναι ορατά ως modules του μοντέλου) και επίσης με αυτόν τον τρόπο γίνεται και αυτόματη αρχικοποίηση των βαρών (**weights**) του κάθε επιπέδου με βάση την activation function που του αντιστοιχεί αφού παρατηρήθηκε στην αρχή με κάποιες δοκιμές πως χωρίς αυτή την αρχικοποίηση σε αρκετές περιπτώσεις τα βάρη ενημερώνονταν με πάρα πολύ μικρό ρυθμό. Το τελευταίο ήταν μια πολύ καλή ένδειξη του προβλήματος των **vanishing gradients**. Οπότε σύμφωνα με την βιβλιογραφία στην περίπτωση της **ReLU** γίνεται αρχικοποίηση ομοιόμορφης κατανομής του **HE**, στην περίπτωση της **SELU** γίνεται αρχικοποίηση ομοιόμορφης κατανομής του **LeCun** (default) και στην περίπτωση της **Softmax** γίνεται αρχικοποίηση ομοιόμορφης κατανομής του **Glorot/Xavier**. Τέλος, δίνεται και η επιλογή της εισαγωγής ενός επιπέδου στην αρχή για batch normalization της εισόδου για ένα απλό scaling ανά batch (αντί του standardization σε όλο το dataset).

- **Forward pass** (forward): Με βάση την σειρά δήλωσης των επιπέδων δίνεται η είσοδος στο πρώτο (αν έχει δηλωθεί batch normalization τότε περνάει από το αντίστοιχο αυτό επίπεδο) και αφού γίνουν όλοι οι μετασχηματισμοί εφαρμόζεται στο αποτέλεσμα του το αντίστοιχο activation function (αν έχει δηλωθεί) και αυτή η διαδικασία επαναλαμβάνεται και συνεχίζεται μέχρι και το τελευταίο αφού κάθε επίπεδο λαμβάνει την είσοδο από το προηγούμενο.
- **Training ανά batch** (batchTrain): Πραγματοποιείται όλη η λογική του training μέσα σε μια επανάληψη (**epoch**) όπου σε κάθε batch γίνεται ένα **forward** και **backward pass** με τον συνηθισμένο τρόπο που ακολουθείται στο pytorch. Επειδή κάποιοι scheldulers ενημερώνουν το learning rate στο τέλος του κάθε batch και όχι στο τέλος του κάθε epoch (One Cycle) γίνεται και αυτός ο έλεγχος ενημέρωσης. Τέλος, υπολογίζεται το **μέσο loss** που προκύπτει από όλα τα batches, μια πιο ακριβή προσέγγιση για τον υπολογισμό της **μέσης ακρίβειας** δηλαδή του accuracy score και μια πιο προσεγγιστική τιμή για το **μέσο f1 score**.
- **Evaluation ανά batch** (batchEval): Παρόμοια λογική με το batchTrain αλλά τώρα χρησιμοποιείται μόνο για την εξαγωγή στατιστικών στοιχείων επίδοσης του μοντέλου από το forward pass.
- **Training** (trainNN): Διεξάγεται όλη η διαδικασία του training χωρίζοντας αρχικά όλο το training set σε **batches** και στην συνέχεια χρησιμοποιεί την batchTrain σε κάθε epoch. Επίσης αν υπάρχει schelduler ελέγχει αν θα πρέπει να δώσει την ευθύνη χρήσης του στο batchTrain ή αν θα πρέπει να τον επικαλείται στο τέλος του κάθε epoch.
- **Training με επιλογές** (trainNNWithOptions): Παρόμοια λογική με το απλό trainNN απλά η κύρια διαφορά είναι πως τώρα γίνεται η χρήση του

validation set για συμβουλευτικό κυρίως λόγο οπότε θα πρέπει να χωριστεί και αυτό σε **batches** έτσι ώστε να μετριούνται έπειτα από κάθε batchTrain του ίδιου epoch τα scores πάνω στα batches αυτά του validation. Έτσι όλα μαζί αυτά τα scores μπορούν να χρησιμοποιηθούν στην σχεδίαση των **learning curves** αλλά και κάθε ένα score (loss score) βοηθάει στον εντοπισμό του **overfitting.** Οπότε αν αρχίζει το validation loss να αυξάνεται μπορεί να εφαρμοστεί μετά από έναν μικρό αριθμό επαναλήψεων (**patience**) που παρατηρείται αυτό το φαινόμενο η άμεση διακοπή της διαδικασίας του training (**early stopping**).

- **Προβλέψεις** (predict): Το εκπαιδευμένο μοντέλο μπορεί να παραγάγει προβλέψεις για οποιαδήποτε dataset ή instance που δέχεται ως είσοδο και είτε φέρνει την **κατανομή πιθανότητας** είτε τις **προβλέψεις των κλάσεων**.

  Κλείνοντας πριν τεθεί το μοντέλο σε εφαρμογή ορίζονται κάποιες συμπληρωματικές συναρτήσεις που εξυπηρετούν συγκεκριμένους σκοπούς.

  Αρχικά, ορίζεται μια συνάρτηση για τον υπολογισμό των **roc curves**. Η υλοποίηση βασίστηκε κυρίως στον τρόπο που προτείνει η sklearn βιβλιοθήκη όπου ο υπολογισμός γίνεται σε δύο στάδια. Για αρχή θα πρέπει να μετατραπούν σε **one-hot vectors** τα labels (αντί για 0, 1 και 2 να υπάρχει 1 στο αντίστοιχο index και 0 στα υπόλοιπα σε ένα διάνυσμα μεγέθους 3 στην περίπτωση αυτή). Σε δεύτερο στάδιο, υπολογίζεται **για κάθε κλάση** το **roc curve** και το **roc auc** (ο χώρος κάτω από την καμπύλη που καθορίζει τον βαθμό τον οποίο το μοντέλο μπορεί εύκολα να αναγνωρίσει/διαχωρίσει τη κάθε κλάση) ως προς τις άλλες δύο κλάσεις (**one vs rest**). Ακόμα, υπολογίζεται και το roc curve και το roc auc για τον μέσο όρο όλων των κλάσεων μαζί με δύο διαφορετικές προσεγγίσεις (micro & macro average).

  Επιπλέον, υπάρχει μια συνάρτηση που είναι υπεύθυνη για την **αξιολόγηση** του μοντέλου σε όλο το test/validation set τυπώνοντας τα **scores** και τα **confusion matrices** (για κάθε κλάση χωριστά και για όλες τις κλάσεις μαζί) όπως στην προηγούμενη εργασία προσθέτοντας ταυτόχρονα στα αποτελέσματα και τα **roc curves**.

  Τέλος, η πιο σημαντική συνάρτηση είναι αυτή του custom **grid search** όπου γράφοντας με έναν πιο προσιτό τρόπο ένα grid με όλες τις διαφορετικές επιλογές σε μια **λίστα** για κάθε οντότητα (π.χ μια λίστα με optimizers, επίπεδα, loss functions …) και ρυθμίζοντας και κάποιες άλλες παραμέτρους προς εξέταση (batch size, epochs …) θα παραχθούν όλοι οι δυνατοί συνδυασμοί και θα δημιουργηθούν με αυτόματο τρόπο διάφορα μοντέλα προς εξέταση. Επίσης, μέσα στη συνάρτηση αποτυπώνονται κάποιες

προκαθορισμένες αποφάσεις υλοποίησης όπου μια εκ των οποίων είναι πως στο loss function χρησιμοποιείται το **άθροισμα** (reduction = sum) αντί για τον μέσο όρο (reduction = mean) λάθους όλων των batches μιας και από δοκιμές που έγιναν παρατηρήθηκε πως συνδυαζόταν καλύτερα με τους διάφορους scheldulers οι οποίοι κάνουν έναν αρκετά μεγάλο αριθμό από βήματα τροποποίησης του learning rate (το learning rate βέβαια θα πρέπει να είναι μικρό ώστε να μπορεί να αντισταθμίσει την μεγάλη τιμή του loss λόγω του αθροίσματος). Ακόμα, κάποιες άλλες αποφάσεις που πάρθηκαν αφορούν τις **τιμές** που δόθηκαν ως παράμετροι στους **scheldulers** οι οποίοι θα χρησιμοποιηθούν με τους μη adaptive optimizers. Σε τελευταίο στάδιο, αφού πραγματοποιηθεί η εκπαίδευση και η αξιολόγηση του κάθε μοντέλου αντίστοιχα προστέθηκε μια **διαισθητικά** καλή προσέγγιση για τον υπολογισμό των **καλών μοντέλων** η οποία κρατάει το μοντέλο με το μεγαλύτερο **λόγο accuracy score** προς την **κανονικοποιημένη ευκλείδεια απόσταση** (μοίαζει με την mse loss) μεταξύ των σημείων των **καμπύλων** του **training** και του **validation loss** (κανονικοποιημένη μιας και το σύνολο των σημείων δεν είναι το ίδιο σε κάθε γραφική παράσταση λόγω early stopping) οπότε διαισθητικά σημαίνει πως αυτό το μοντέλο που θα χει αρκετά μεγάλο accuracy score και μικρή απόσταση μεταξύ των γραφικών παραστάσεων θα είναι προσεγγιστικά ένα καλό υποψήφιο μοντέλο. Όπως θα παρατηρηθεί και παρακάτω υπάρχει ένα **trade off** μεταξύ του καλού accuracy score και των καλών καμπυλών και επειδή στόχος είναι ένα πιο υψηλό accuracy score ώστε να αυξηθούν και οι υπόλοιπες μετρικές (f1, precision, recall) και κυρίως να αποφευχθεί το underfitting τίθεται ένα accuracy threshold (π.χ > 70 %) και διερευνούνται καμπύλες οι οποίες είναι καλές άλλα με πιθανώς κάποιο τίμημα στην μεταξύ τους απόσταση. Παρόλα αυτά μια γρήγορη ματιά μπορεί να επαληθεύσει την προσέγγιση αυτή είτε μπορεί και να βρεθεί κάτι καλύτερο όμως παρόλα αυτά δίνει μια διαισθητική προσέγγιση της εικόνας των καμπυλών για το συγκεκριμένο threshold.

Μοντέλα που προκύπτουν από 2 Διαφορετικές Προσεγγίσεις

TF-IDF προσέγγιση: Σε αυτή την προσέγγιση θα δοκιμαστεί το κάθε μοντέλο λαμβάνοντας ως είσοδο την TF-IDF απεικόνιση για τα κείμενα των tweets με τη χρήση του TfidfVectorizer. Η διαδικασία για την απεικόνιση αυτή είναι παρόμοια με αυτή της προηγούμενης εργασίας με την μόνη διαφορά πως τώρα θα πρέπει να γίνει μετατροπή της απεικόνισης σε **tensors** και πως τώρα το **λεξιλόγιο** είναι **αρκετά μικρότερο** (3000 λέξεις) επειδή η διαδικασία της εκπαίδευσης είναι αρκετά πιο χρονοβόρα.

GLoVe προσέγγιση: Σε αυτή την προσέγγιση θα δοκιμαστεί το κάθε μοντέλο λαμβάνοντας ως είσοδο την απεικόνιση των κειμένων με την χρήση των word vectors/embeddings της κάθε λέξης που απαρτίζεται το καθένα. Για τον σκοπό αυτόν θα χρησιμοποιηθούν τα προ-εκπαιδευμένα (**pretrained**) word vectors πάνω σε **tweets** που παρέχει το **GLoVe**. Ως πρώτο βήμα, λοίπον, κατεβάζονται **τοπικά** στο colab όλα τα σχετικά αρχεία με τα embeddings όμως το μόνο που θα χρησιμοποιηθεί είναι αυτό με τις **100 διαστάσεις** μιας και δοκιμάστηκε έναντι των υπολοίπων και έδειξε καλύτερα αποτελέσματα. Το επόμενο βήμα είναι να δημιουργηθεί το αντίστοιχο **λεξιλόγιο** και αυτό επιτυγχάνεται με την χρήση μιας αντιστοιχίας (**mapping**) της κάθε λέξης του λεξιλογίου με τον αριθμό (**index**) της αντίστοιχης γραμμής που βρίσκεται το διάνυσμα (vector) της λέξης αυτής σε έναν μεγάλο πίνακα που δημιουργείται από όλα τα διανύσματα. Με αυτόν τον τρόπο διευκολύνονται οι πράξεις που αφορούν κάποια σημεία αργότερα όπου απαιτείται η συγχώνευση (**aggregation**) τέτοιων διανυσμάτων. Οπότε διαβάζεται **γραμμή-γραμμή** το αντίστοιχο αρχείο και σε κάθε τέτοια γραμμή που αφορά μια λέξη διαχωρίζεται η λέξη αυτή από το αντίστοιχο διάνυσμα της και δημιουργείται η παραπάνω αντιστοιχία. Στο τέλος, προστίθεται ένα διάνυσμα το οποίο αποτελείται από τον **μέσο όρο** όλων των υπολοίπων του λεξιλογίου ώστε να αντιστοιχίζονται όλες οι λέξεις που δεν ανήκουν στο λεξιλόγιο (μια καλή λύση που είχε προταθεί σε προηγούμενο μάθημα αντί για το μηδενικό διάνυσμα). Αφού το λεξιλόγιο έχει δημιουργηθεί για κάθε κείμενο παράγεται το **μέσο διάνυσμα** από τα διανύσματα των λέξεων που αποτελείται (το οποίο σε αυτή τη φάση **δεν αποτελεί** και την καλύτερη λύση αλλά δεν υπάρχει και κάποια καλύτερη επιλογή).

Αξιολόγηση των μοντέλων για κάθε μια από τις παραπάνω προσεγγίσεις

Η αξιολόγηση των μοντέλων και για τις δύο προσεγγίσεις θα γίνει ως προς τις διαφορετικές υπερπαραμέτρους των μοντέλων στις εξής κατηγορίες:
1) Αριθμός επιπέδων (layers) και νευρώνων (neurons) σε κάθε επίπεδο
2) Activation functions
3) Optimizers
4) Loss functions
5) Συνδυασμός όλων των παραπάνω (μιας και όλα **αλληλοεξαρτώνται**)

Στις περιπτώσεις 1-4 δοκιμάζονται διαφορετικές οντότητες της αντίστοιχης κατηγορίας για διαφορετικά learning rates (κάποια με schelduler και κάποια χωρίς) κρατώντας σταθερά μια οντότητα από κάθε άλλη κατηγορία (επιλέγονται κάθε φορά κάποιες συνηθισμένες). Για παράδειγμα, η

περίπτωση 1 έχει μια σταθερή activation function που είναι η ReLU, έναν σταθερό optimizer που είναι ο κλασσικός SGD και μια loss function που είναι η Cross Entropy.

Οι περιπτώσεις 2-5 ελέγχονται **μόνο** ως προς τα δύο καλύτερα σε αριθμό επίπεδα (1, 2 ή 3 επίπεδα) και αντίστοιχα με τον καλύτερο αριθμό νευρώνων.

Σε όλες τις περιπτώσεις όσον αφορά τις **τιμές** κάποιων άλλων παραμέτρων όπως για το batch size, τα epochs, το early stopping patience και το L2 regularization παραμένουν σταθερές έπειτα από αρκετές δοκιμές πως ο συνδυασμός τους δουλεύει γενικά καλά.
Επίσης, σε όλες τις περιπτώσεις τυπώνονται τα αποτελέσματα του κάθε μοντέλου μαζί με τα learning curves για την πιο αποτελεσματική σύγκριση τους.

(Οι συγκρίσεις που θα παρουσιαστούν στην συνέχεια έχουν να κάνουν με τα συγκεκριμένα στιγμιότυπα που θα γίνουν οι δοκιμές για να εξαχθούν κάποιες χρήσιμες πληροφορίες σχετικά με την συμπεριφορά των διαφορετικών παραμέτρων. Οπότε κυρίως θα συγκριθούν οι ποιότητες των διαφορετικών learning curves που παράγονται και λιγότερο τα scores (θα γίνεται αναφορά στο μέσο score) εφόσον τα learning curves δεν αλλάζουν αρκετά από εκτέλεση σε εκτέλεση αλλά τα scores μπορεί να αλλάξουν. Βέβαια, στο τέλος συνδυάζονται όλες οι περιπτώσεις αφού τα πάντα αλληλοεξαρτώνται όπως αναφέρθηκε και παραπάνω.)

Για τις παρακάτω προσεγγίσεις έχουν δημιουργηθεί δύο διαφορετικά .ipynb αρχεία μιας και το μέγεθος θα ήταν αρκετά μεγάλο αν βρίσκονταν και οι δύο μόνο σε ένα.

TF-IDF προσέγγιση (AI2_HW2_TFIDF.ipynb)

Σταθερές παράμετροι:
- epochs : 50
- batch size : 32
- early stopping patience : 3
- L2 regularization : 0.1 (χρειάζεται αρκετό regularization για την αποφυγή του overfitting)
- Χωρίς batch normalization της εισόδου

1) Αριθμός επιπέδων και νευρώνων σε κάθε επίπεδο

Στην περίπτωση αυτή συγκρίνονται:

a) i) 1 hidden layer με 75 νευρώνες (μικρός αριθμός) και
   ii) 1 hidden layer με 675 νευρώνες (μεγάλος αριθμός)
b) i) 2 hidden layers με 220 και 90 νευρώνες αντίστοιχα (μικρός αριθμός) και
   ii) 2 hidden layers με 660 και 270 νευρώνες αντίστοιχα (μεγάλος αριθμός)
c) i) 3 hidden layers με 230, 120 και 50 νευρώνες αντίστοιχα (μικρός αριθμός) και
   ii) 3 hidden layers με 690, 360 και 150 νευρώνες αντίστοιχα (μεγάλος αριθμός)

Στα αποτελέσματα του a διαφέρει κυρίως μόνο ο **χρόνος** της εκπαίδευσης μεταξύ των δύο περιπτώσεων (καλύτερος με λιγότερους νευρώνες) που συγκρίνονται (ai και aii) για όλα τα διαφορετικά learning rates και παρατηρείται και ένα **μικρό overfitting** σε κάθε περίπτωση. Ομοίως και για το b και για το c.

Αυτό που παρατηρείται επίσης σε μεγάλο βαθμό είναι πως **όσο αυξάνεται** ο αριθμός των hidden layers τόσο τα αποτελέσματα των **learning curves γίνονται χειρότερα** και ο βαθμός του **overfitting μεγαλώνει** ειδικά στις περιπτώσεις με 3 layers (ci και cii).

Σύμφωνα με τα παραπάνω θα χρησιμοποιηθούν στις **επόμενες συγκρίσεις** τα επίπεδα και οι νευρώνες που περιγράφονται στο ai και bi.

2) Activation functions

Στην περίπτωση αυτή συγκρίνονται:

a. ReLU
b. SELU

Τα αποτελέσματα μεταξύ της ReLU και της SELU διαφέρουν κυρίως ως προς τα learning curves όπου για τη ReLU είναι αρκετά καλύτερα και διακρίνεται αρκετά **μικρότερο overfitting** (η SELU κάνει πολύ πιο γρήγορα early stopping).

3) Optimizers

Στην περίπτωση αυτή συγκρίνονται:

a. SGD
b. Nesterov
c. Adadelta
d. Adam.

Θέλει λίγο **προσοχή** με τις **παραμέτρους** στον Adam και στον Adadelta γιατί με τις συγκεκριμένες παραμέτρους φαίνεται ότι μπορεί να πέσουν **εύκολα** σε **underfitting**. Σε διαφορετική περίπτωση ο Adadelta καταφέρνει το **καλύτερο fit** με **χαμηλό** βέβαια **score** ενώ ο Adam πετυχαίνει **μέτριο fit** με **καλό score**. Τα **learning curves** του SGD είναι **πιο ομαλά** και σταθερά σε αντίθεση με του Nesterov αλλά ο Nesterov μπορεί να καταφέρει **καλύτερα scores** κυρίως με λιγότερα επίπεδα. Τέλος, οι SGD και Nesterov είναι **πιο γρήγοροι** μιας και δεν έχουν να κάνουν πολλούς περίπλοκους υπολογισμούς.

4) Loss functions

Στην περίπτωση αυτή συγκρίνονται:

a. Cross Entropy
b. Hinge Multinomial

Και στις δύο περιπτώσεις τα **learning curves** είναι καλά με έναν **μικρό** βαθμό από **overfitting**. Επιβεβαιώνεται ωστόσο πως η Cross Entropy λειτουργεί καλά σε datasets τα οποία **δεν είναι ισορροπημένα** (κυρίως για τη κλάση antivax) μιας και προβλέπει καλά όλες τις κλάσεις (και την antivax) έχοντας έτσι καλύτερα scores από την Hinge Multinomial.

5) Από τον συνδυασμό των παραπάνω παραμέτρων παράχθηκαν **96 συνολικά μοντέλα**. Από αυτά κάποια ξεχώρισαν για το μεγάλα scores τους αλλά και για τα καλά learning curves τους. Αυτά τα μοντέλα αντιστοιχούσαν στα grids με αριθμό 13, 44, 69, 72 και 85 (υπολογίστηκε πως ήταν καλό μοντέλο και με τον διαισθητικό τρόπο όπως φαίνεται και στο τέλος).

Σε αυτά τα μοντέλα λοιπόν το καλύτερο μοντέλο ήταν το μοντέλο του grid 13 (βέβαια με κάποια αύξηση του αριθμού των epochs από 50, που είχε τεθεί μικρό λόγω των μεγάλων διαστάσεων, σε 200) το οποίο είχε:

- 1 επίπεδο με 75 νευρώνες
- Optimizer: Adadelta
- Activation function: SELU

- Loss function: Cross Entropy
- Χωρίς schelduler
- Learning rate: 0.01

Το μοντέλο αυτό είχε αρκετά ικανοποιητική επίδοση μιας και έχει καταφέρει περίπου **71.5% accuracy** και **70% f1 score**. Βέβαια, υπάρχουν και υψηλότερα scores κοντά στο 73.5% αλλά το συγκεκριμένο μοντέλο είναι αυτό που έχει **τα καλύτερα learning curves** τόσο για accuracy όσο και για loss και για τα οποία παρόλο τον μεγάλο αριθμό των epochs οι καμπύλες του training με αυτές του validation διατηρούνται **αρκετά κοντά** κρατώντας τον βαθμό του **overfitting σχεδόν μηδαμινό**.

Όσον αφορά τα roc curves της κάθε κλάσης παρατηρείται πως απέχουν αρκετά **μακριά** από την **διαγώνιο** οπότε είναι μια καλή ένδειξη ότι οι προβλέψεις δεν είναι τελείως τυχαίες. Βέβαια, δεν αποτελούν και τις καλύτερες καμπύλες εφόσον απέχουν αρκετά από την πάνω αριστερά γωνία. Ωστόσο, σύμφωνα με το roc auc score καταλαβαίνει κανείς πως η κλάση neutral που έχει το μεγαλύτερο auc score (87%) μπορεί **να διακριθεί πιο εύκολα** από το μοντέλο από τις μη neutral (antivax + provax μαζί) ενώ αντίστοιχα στη συνέχεια το ίδιο ισχύει σε έναν μικρότερο βαθμό για την antivax κλάση σε σχέση με τις μη antivax (neutral + provax μαζί) και σε ακόμα πιο μικρό βαθμό για την provax κλάση σε σχέση με τις μη provax (neutral + antivax μαζί), χωρίς όμως να πέφτει γενικά το auc score κάτω από το 84% που σημαίνει πως οι προβλέψεις της κάθε κλάσης για άγνωστες οντότητες είναι αρκετά καλές.

GLoVe προσέγγιση (AI2_HW2_GLoVe.ipynb)

Σταθερές παράμετροι:
- epochs : 100
- batch size : 32
- early stopping patience : 3
- L2 regularization : 10^{-4} (χρειάζεται λίγο regularization για την αντιμετώπιση του overfitting)
- Με απλό batch normalization της εισόδου

1) Αριθμός επιπέδων και νευρώνων σε κάθε επίπεδο

Στην περίπτωση αυτή συγκρίνονται:

a) i) 1 hidden layer με 75 νευρώνες (μικρός αριθμός) και
ii) 1 hidden layer με 675 νευρώνες (μεγάλος αριθμός)

b) i) 2 hidden layers με 220 και 90 νευρώνες αντίστοιχα (μικρός αριθμός) και
ii) 2 hidden layers με 660 και 270 νευρώνες αντίστοιχα (μεγάλος αριθμός)
c) i) 3 hidden layers με 230, 120 και 50 νευρώνες αντίστοιχα (μικρός αριθμός) και
ii) 3 hidden layers με 690, 360 και 150 νευρώνες αντίστοιχα (μεγάλος αριθμός)

Στο a αξίζει να σημειωθεί πως το ai έχει τις περισσότερες φορές παρόμοια scores με το aii αλλά αρκετά **καλύτερα learning curves** από το aii τα οποία είναι πιο **ομαλά** με **λιγότερο** βαθμό **overfitting**. Ομοίως και για το b και για το c. Οπότε ο **αριθμός των νευρώνων** παίζει αρκετά **μεγάλο ρόλο**.

Παρατηρείται και εδώ επίσης σε μικρότερο βαθμό πως όσο **αυξάνεται** ο αριθμός των **hidden layers** τα αποτελέσματα των **learning curves** γίνονται λίγο **χειρότερα** από πλευράς **overfitting**.

Γι αυτό στην συνέχεια θα χρησιμοποιηθούν και εδώ στις επόμενες συγκρίσεις το ai και bi.

2) Activation functions

Στην περίπτωση αυτή συγκρίνονται:

a. ReLU
b. SELU

Και στις δύο περιπτώσεις τα αποτελέσματα δεν διαφέρουν πολύ και είναι αρκετά καλά. Γενικά η SELU παράγει **καλά learning curves** με **ελάχιστο overfitting**. Η ReLU έχει λίγο παρόμοια ή και **λίγο χειρότερα learning curves** από την SELU αλλά βγάζει κατά πλειοψηφία **λίγο καλύτερα scores**.

3) Optimizers

Στην περίπτωση αυτή συγκρίνονται:

a. SGD
b. Nesterov
c. Adadelta
d. Adam.

Ο SGD έχει σχεδόν **τα καλύτερα learning curves** με το **λιγότερο overfitting** και με αρκετά καλά scores και στην συνέχεια ακολουθεί ο Nesterov. Βέβαια και εδώ θέλει αρκετή **προσοχή** με τις **παραμέτρους** στον Adam και στον Adadelta αφού μπορούν να υποστούν **underfitting**. Ωστόσο, για τις συγκεκριμένες παραμέτρους όταν καταφέρνουν **καλύτερο fit** έχουν **πολύ χαμηλά scores**.

4) Loss functions

Στην περίπτωση αυτή συγκρίνονται:

a. Cross Entropy
b. Hinge Multinomial

Και στις δύο περιπτώσεις τα **learning curves** είναι καλά με έναν **μικρό** βαθμό από **overfitting** (λίγο λιγότερο στην Hinge Multinomial). Επιβεβαιώνεται και εδώ ωστόσο όπως και στην προηγούμενη προσέγγιση πως η Cross Entropy λειτουργεί καλά σε datasets τα οποία **δεν είναι ισορροπημένα** (κυρίως για τη κλάση antivax) μιας και προβλέπει καλά όλες τις κλάσεις (και την antivax) έχοντας έτσι **καλύτερα scores** από την Hinge Multinomial.

5) Συνδυασμός όλων των παραπάνω

Από τον συνδυασμό των παραπάνω παραμέτρων παράχθηκαν **96 συνολικά μοντέλα**. Από αυτά κάποια ξεχώρισαν για τα μεγάλα scores τους αλλά και για τα καλά learning curves τους. Αυτά τα μοντέλα αντιστοιχούσαν στα grids με αριθμό 3, 20 (υπολογίστηκε πως ήταν καλό μοντέλο και με τον διαισθητικό τρόπο όπως φαίνεται και στο τέλος), 36 και 57.

Σε αυτά τα μοντέλα λοιπόν το καλύτερο μοντέλο ήταν το μοντέλο του grid 3 το οποίο είχε:

- 1 επίπεδο με 75 νευρώνες
- Optimizer: SGD/Momentum (λόγω του One Cycle)
- Activation function: SELU
- Loss function: Cross Entropy
- Schelduler: One Cycle
- Learning rate: $5*10^{-7}$

Το αποτέλεσμα του μοντέλου αυτού ξεπέρασε στην συγκεκριμένη εκτέλεση το **70%** του accuracy διατηρώντας υψηλά στο **69%** και το **f1 score**. Γενικότερα είναι αρκετά καλά scores εφόσον **δεν έχουν ξαναβρεθεί** πουθενά αλλού στις δοκιμές του GLoVe τόσο υψηλά. Επίσης, όπως φαίνεται και στα learning curves δεν υπάρχει καθόλου underfitting και ούτε αρκετό

overfitting μιας και βρίσκονται αρκετά κοντά οι καμπύλες του training και του validation.

Όσον αφορά τα roc curves ισχύουν παρόμοια συμπεράσματα με την προσέγγιση του TF-IDF βέβαια σε αυτή την περίπτωση οι καμπύλες είναι **πιο χαμηλές** προς την διαγώνιο και προφανώς και το auc score της κάθε κλάσης.

Καλύτερο μοντέλο

Συγκρίνοντας τα καλύτερα μοντέλα των παραπάνω δύο διαφορετικών προσεγγίσεων το **καλύτερο γενικά** μοντέλο είναι αυτό της **TF-IDF προσέγγισης** λόγω της αρκετά **καλύτερης ποιότητας** που προκύπτει από τα **learning curves** για accuracy και loss (**λιγότερο overfitting** από την GLoVe προσέγγιση) και λόγω των **καλύτερων αποτελεσμάτων** στα **τελικά scores** (accuracy, f1 , precision, recall) και στα roc curves και πιο συγκεκριμένα στα **roc auc scores** της κάθε κλάσης (**2-3%** καλύτερα από αυτά της GLoVe προσέγγισης).

Σύγκριση με το καλύτερο μοντέλο της Εργασίας 1

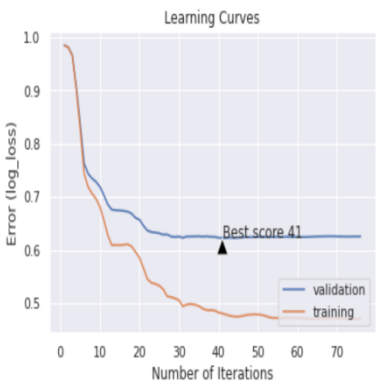
Softmax Regression

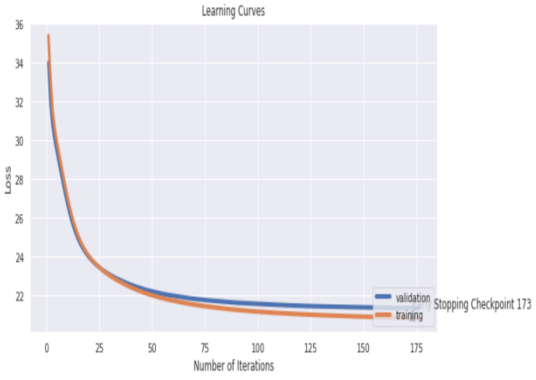
	precision	recall	f1-score
neutral	0.78525	0.80000	0.79256
antivax	0.71739	0.44595	0.55000
provax	0.69694	0.76656	0.73009
accuracy			0.74058
macro avg	0.73319	0.67083	0.69088
weighted avg	0.74081	0.74058	0.73589

Feed Forward Neural Network

	precision	recall	f1-score
neutral	0.764759	0.790610	0.777470
antivax	0.701493	0.317568	0.437209
provax	0.664756	0.755700	0.707317
macro avg	0.710336	0.621293	0.640665
weighted avg	0.716192	0.715162	0.705021

Accuracy: 0.71516





Όπως φαίνεται από τα παραπάνω το **softmax regression** μοντέλο έχει αρκετά **καλύτερα τελικά scores** (accuracy, f1 , precision, recall) από το feed forward νευρωνικό. Βέβαια, αν παρατηρήσει κανείς τα **learning curves** παρατηρείται μια **τεράστια διαφορά** ανάμεσα τους. Στην περίπτωση του **softmax regression** υπάρχει αρκετά **μεγάλο overfitting** το οποίο ήταν αρκετά δύσκολο να επιλυθεί μιας και το **εύρος** των διαθέσιμων **υπερπαραμέτρων** ήταν πιο **περιορισμένο** σε σύγκριση με τώρα. Το μοντέλο δηλαδή του **feed forward** φαίνεται να κάνει **άριστη γενίκευση** (generalization) σε άγνωστα δεδομένα σε σύγκριση με το μοντέλο του softmax regression. Κλείνοντας, αξίζει να επισημανθεί πως το **softmax regression** μοντέλο είχε εκπαιδευτεί με **πολύ μεγαλύτερο λεξιλόγιο** (μεγέθους 12000) περιλαμβάνοντας unigrams και bigrams και αυτό το γεγονός το κάνει να παράγει καλύτερα scores χωρίς όμως να μπορεί να ανταγωνιστεί τα learning curves του feed forward μοντέλου αντιμετωπίζοντας τόσο αποτελεσματικά το overfitting.

Οδηγίες για τη δοκιμή με το test set

Το καλύτερο μοντέλο βρίσκεται στο αρχείο **AI2_HW2_TFIDF.ipynb**. Για να εξεταστεί το test set θα πρέπει αρχικά να φορτωθεί με το όνομα του validation set ή να δοθεί το σωστό μονοπάτι. Θα πρέπει επίσης να εκτελεστούν όλα τα κελία από την αρχή χωρίς όμως την υποπεριοχή που διεξάγονται όλα τα διαφορετικά πειράματα 1-5 που περιγράφηκαν παραπάνω (Model Evaluation & Tuning Section) μιας και διαρκούν αρκετή ώρα και χρησιμοποιούσαν το validation set.

Βιβλιογραφία/Πηγές

- Διαφάνειες μαθήματος
- Pytorch documentation (tutorials + examples)
- Pytorch discussions
- Sklearn documentation (tutorials + examples)
- Stack Overflow
- Stack Exchange
- Youtube tutorials:
  https://www.youtube.com/watch?v=oPhxf2fXHkQ
  https://www.youtube.com/watch?v=mM6apVBXGEA
  https://www.youtube.com/watch?v=xWQ-p_o0Uik
  https://www.youtube.com/watch?v=81NJgoR5RfY
  https://www.youtube.com/watch?v=4jRBRDbJemM
  https://www.youtube.com/watch?v=p3CcfIjycBA
  …

- Πάρα πολλά άρθρα, blogs και forums για ιδέες πάνω στο θέμα όπως:
  https://towardsdatascience.com/useful-plots-to-diagnose-your-neural-network-521907fa2f45
  https://androidkt.com/calculate-total-loss-and-accuracy-at-every-epoch-and-plot-using-matplotlib-in-pytorch/
  https://neptune.ai/blog/how-to-choose-a-learning-rate-scheduler
  https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/
  https://www.researchgate.net/post/What-is-the-value-of-the-area-under-the-roc-curve-AUC-to-conclude-that-a-classifier-is-excellent
  https://www.machinecurve.com/index.php/2021/07/19/how-to-use-pytorch-loss-functions/
  https://ruder.io/optimizing-gradient-descent/
  …

Ορφέας Τσουράκης,
1115201700175