Εξόρυξη Δεδομένων κ' Αλγόριθμοι Μάθησης

Εργαστηριακή άσκηση '19-'20

Βλασσόπουλος Γιάννης – Α.Μ. 5486

- **Περιβάλλον Υλοποίησης**

Για τα ερωτήματα 1-2 χρησιμοποίησα conda περιβάλλον. Σε Linux μπορεί να εγκατασταθεί από το default package manager στα περισσότερα distros, για windows το executable installer μπορεί να βρεθεί στο: https://docs.conda.io/projects/conda/en/latest/user-guide/install/windows.html

Για το ερώτημα 1 χρησιμοποιήθηκαν οι βιβλιοθήκες pandas και NumPy για ευκολία στο χειρισμό των δεδομένων. Για το SVM, k-means και scalers χρησιμοποιήθηκε η scikit-learn.

Για το ερώτημα 2 επιπλέον χρησιμοποιήθηκε η nltk και για το κομμάτι των νευρωσικών, pytorch (απλά επειδή μ αυτή είμαι familiar).

Χρησιμοποιήθηκε επίσης ο ~~Ranger optimizer (από GitHub, αναφέρεται παρακάτω) μιας και έχει φανεί να συμπεριφέρεται καλυτέρα απ' τον Adam (και αντίστοιχα με Adam + warmup) (https://arxiv.org/pdf/2004.01461v2.pdf)~~ Adam optimizer. Στο συγκεκριμένο αντικείμενο δεν είδα καμία διαφορά μεταξύ των 2 optimizers οπότε χρησιμοποίησα απλά τον Adam για να μη σας παιδεύω στην αναπαραγωγή του περιβάλλοντος(, μιας και ο Ranger δεν υπάρχει σε pip η conda repos και χρειάζεται εγκατάσταση από git).
Επίσης χρησιμοποιήθηκε Mish activation (https://arxiv.org/ftp/arxiv/papers/1908/1908.08681.pdf). Ο κώδικας περιλαμβάνεται.

Για την εγκατάσταση των παραπάνω, σε περιβάλλον conda μπορούν να εκτελεστούν οι παρακάτω εντολές:

conda install pandas NumPy nltk scikit-learn
conda install pytorch cudatoolkit=10.2 -c pytorch

- **Υλοποίηση**

Ερώτημα 1.

Φορτώνω αρχικά το αρχείο csv και χωρίζω σε 2 πίνακες, train και validation. Κάνω train στο αρχικό training set, και ελέγχω μετά την ακρίβεια του μοντέλου. Το shuffle, spilt και training – validation γίνεται 20 φορές για normalization (ελέγχεται από τη μεταβλητή 'tries')

Μετά δημιουργώ νέες στήλες 'pH' όπως αναφέρουν τα υποερώτηματα. Πετάμε το 1/3 και οι τιμές συμπληρώνονται με το mean η k-means. Φτιάχνονται νέα μοντέλα, 1 παραλείποντας τελείως τη στήλη 'pH' και τα υπόλοιπα χρησιμοποιούν τις καινούργιες και μετρά ελέγχονται στα δεδομένα του test set του οποίου οι τιμές δεν έχουν πειραχτεί. Και αυτή η διαδικασία γίνεται 'tries' φορές για να έχουμε ένα μέσο ορό ακρίβειας.

Ερώτημα 2.

Φορτώνονται τα δεδομένα αρχικά σ' ένα pandas DataFrame. Έχουμε 4 συναρτήσεις οι οποίες λειτουργούν στις στήλες του πίνακα και δημιουργούν νέες στήλες.

Η 1η παίρνει τα raw text δεδομένα της πρώτης στήλης και κάνει το stemming/ cleanup/ χωρίζει σε tokens και επιστρέφει ένα array με τα tokens αυτά. Η 2η καταγραφεί τα tf και δημιουργεί το λεξικό μαζί με τα df. Επιστρέφει arrays με τα tf. Η 3η χρησιμοποιώντας τις tokenized προτάσεις και τα tf δημιουργεί το τελικό tf-idf και επιστρέφεται πάλι ως array που αντιστοιχεί στην κάθε πρόταση. Η 4η καταγραφεί απλά το μήκος κάθε πρότασης και κρατάμε το top90% ως αυτό που θα χρησιμοποιηθεί στο δίκτυο, μιας και χρειαζόμαστε input σταθερού μεγέθους.

Το δίκτυο είναι απλό ConvNet, το βάθος του οποίου (#convolutional layers) καθορίζεται από τη μεταβλητή 'depth' και το μήκος φίλτρου από την 'kernel'. Χρησιμοποιούνται 5 κ' 3 αντίστοιχα. Τα βάρη αρχικοποιούνται με βάση το Kaiming initialization scheme (He et al.)

Χρησιμοποιείται BCE loss (binary cross entropy) και ο Adam optimizer με αρχικό learning rate 0.1 και μέγεθος batch είναι 64. Χρησιμοποιείται επίσης scheduler που παρακολουθεί την out-of-sample ακρίβεια του δικτύου και χαμηλώνει το learning rate αν δει ότι η ακρίβεια αυτή δεν ανεβεί μετά από ~3-4 εποχές (εξαρτάται πόσες εποχές είναι το training, ο scheduler περιμένει το 1/6 αυτών)

Εκπαιδεύουμε για 20 εποχές και μετά περνάμε τα δεδομένα testing. Κρατάμε 1 πίνακα με 2 στήλες, 1η τα predicted αποτελέσματα του δικτύου και 2η τις σωστές απαντήσεις που αντιστοιχούν. Με αυτό τον πίνακα φτιάχνουμε και τα τελικά αποτελέσματα f1, recall, precision. Χρησιμοποιώ το >=0.5 ως threshold για να θεωρηθεί το prediction 'Ture'.

- **Αποτελέσματα και σχόλια**

Στο ερώτημα 1 φάνηκε ότι η καλύτερη συμπεριφορά βρίσκεται με τα default settings του SVM της sklearn. Οποιεσδήποτε αλλαγές επέφεραν είτε ιδιά είτε ελαφρά χειρότερη συμπεριφορά. Η raw ακρίβεια του μοντέλου είναι ~62%.
Στο 2ο μέρος οπού πειράζουμε τη στήλη 'pH' βλέπουμε ότι η καλύτερη συμπεριφορά προκύπτει όταν απλά δε συμπεριλάβουμε τη στήλη στο training του SVM, όπου φτάνει πολύ κοντά στην ακρίβεια του 'full' μοντέλου (οριακά ιδιά). Όταν γεμίζουμε τις dropped τιμές με τεχνητές σε κάθε περίπτωση το SVM φέρεται σημαντικά χειροτέρα (~40% ακρίβεια).

Στο 2ο ερώτημα δεν υπήρξαν σημαντικές εκπλήξεις, πέρα απ' το ότι το est των english 'stopwords' του sklearn δεν συμπεριλάμβανε κάποια σημεία στίξης, γι΄ αυτό υπάρχει και η μεταβλητή symbols που έχει εξτρά σημεία στίξης και ελέγχεται μαζί με τα stopwords.
Επίσης στο dataset object φτιάχνω λίγο περίεργα το label tensor. Παρατήρησα ότι όταν έκανα cast το label ως FloatTensor, όταν το label ήταν 0, δημιουργούταν ένα tensor μεγέθους 0, αντί για μεγέθους 1, με τιμή 0. Οπότε δημιουργώ ένα floattensor μεγέθους 1, με τιμή 0, και αν το label είναι 1 απλά προσθέτω 1 στην τιμή του tensor. Όλο αυτό επειδή η BCELoss χρειάζεται floattensor ως arguments.
Τελικά αποτελέσματα:

Ακρίβεια: 76%
Precision: 74%
Recall: 54%
f1: 0.625