Feed Forward Neural Network Project 2

Γεώργιος Κωνσταντίνος Ζαχαρόπουλος sdi1900061

Δεκέμβριος 20, 2022

0.1 Introduction

Στόχος της εργασίας είναι η υλοποίηση ενός sentiment classifier για ένα data set από imdb movie reviews με την χρήση feed forward neural networks.

0.2 Download GloVe Embeddings

Κατεβάζουμε τα GloVe Embeddings τα οποία θα χρησιμοποιηθούν ως είσοδος στο δικό μας μοντέλο.
Μετατρέπουμε το GloVe σε Word2Vec για να έχουμε πιο έυκολη πρόσβαση στα vectors των λέξεων.

0.3 Imports

Κάνουμε import ότι είναι απαραίτητο από τις βιβλιοθήκες: torch, matplotlib, pandas, numpy, re, sklearn, gensim, nltk, torchmetrics

0.4 Loading Data Set

Διαβάζουμε το data set και δημιουργούμε μια στήλη ¨αποτελέσματος' στην οποία βάζουμε 0 αν η κριτική είναι αρνητική και 1 αν είναι θετική. Αυτές θα είναι και οι κλάσεις μας.

0.5 Data Pre-processing

Χρειάζεται να καθαρίσουμε το data set καθώς δεν εκφράζουν όλες οι λέξεις sentiment και αυτό μπορεί να μπερδέψει στο training του μοντέλου.
Για παράδειγμα στα reviews υπάρχουν html tags, αριθμοί που εκφράζουν σκορ, special characters και stop words που δεν προσφέρουν στην εκμάθηση του μοντέλου. Όλα αυτά θα τα αφαιρέσουμε από το data set. Επίσης εφαρμόζουμε lemmatization στις λέξεις του data set. Ετσι μετατρέπουμε τις λέξεις στο αρχικό τους λήμμα, το οποίο θα αυξήσει τις πιθανότητες να βρούμε τα αντίστοιχα vectors στα embeddings. Τέλος αποφεύγουμε την χρήση stemming καθώς μπορεί να ¨χαλάσουν' οι λέξεις και να μην υπάρχει αντιστοιχία με τα embeddings.

0.6 Create the Set of Tensors for the Reviews

Αρχικά μετατρέπουμε το word2vec σε ένα dictionary το οποίο θα μας βοηθήσει να έχουμε γρήγορη αναζήτηση των vectors των λέξων. Επειτα διατρέχουμε τις λέξεις του κάθε review και κρατάμε μόνο τις ξεχωριστές λέξεις σε ένα άλλο dictionary. Ετσι κάθε λέξη θα έχει την ίδια επίδραση στον τελικό tensor του review. Οπότε όταν έχουμε τις ξεχωριστές λέξεις του review βρίσκουμε τους αντίστοιχους tensors και τους προσθέτουμε. Στο τέλος για κάθε review διαιρούμε τον tensor με το πλήθος των ξεχωριστών λέξων του review. Προσθέτουμε τους τελικούς tensors των reviews σε μια λίστα και στο τέλος τους κάνουμε stack.

Σημείωση: Δοκίμασα να χρησιμοποιήσω τον tfidf στα reviews για να βρίσκω τα vectors μόνο των k σημαντικών λέξεων, αλλά δεν είδα βελτίωση στην απόδοση, οπότε δεν το συμπεριέλαβα.

0.7 Split Train and Test Sets

Μετά χωρίζουμε το x και το y σε δύο σύνολα, train και test. Έτσι θα έχουμε ένα άγνωστο για το μοντέλο test set πάνω στο οποίο θα το βαθμολογήσουμε και ένα train set με το οποίο θα εκπαιδεύσουμε το μοντέλο μας. Ετσι ελέγχουμε την απόδοση του και αν κάνει overfit ή undefit.

0.8 Define the Neural Network Class

0.8.1 Base model - 3 Hidden Layers

Αρχικά θα πάρουμε μετρήσεις από το βασικό απλό μοντέλο του φροντηστηρίου που έχει απλά 3 hidden layers με H1=128 H2=64 H3=32, learning rate=1e-4, batch size = 64, χρησιμοποιεί το loss function MSELoss και το optimizer SGD.

Epoch 25	Loss	Accuracy	Precision	Recall	f1
Train Set	8.46204	0.83825	0.83954	0.83962	0.83605
Test Set	8.63679	0.83801	0.81513	0.87416	0.84204

0.8.2 Base model - 2 Hidden Layers

Θα δοκιμάσουμε την επίδοση του μοντέλου με 2 hidden layers με H1=64 H2=16, learning rate=1e-4, batch size = 64, loss function MSELoss και το optimizer SGD.

Epoch 25	Loss	Accuracy	Precision	Recall	f1
Train Set	8.46565	0.83874	0.83899	0.84039	0.83666
Test Set	8.63609	0.83603	0.81695	0.86616	0.83922

Παρατηρούμε ότι δεν υπαρχουν μεγάλες διαφορές, θα κρατήσουμε τα 3 hidden layers όμως γιατί δίνει λίγο μικρότερο loss στο train set και υψηλότερο accuracy.

0.8.3 Base model - Different neuros per hidden layer

Δοκίμασα διαφορετικούς συνδυασμούς νευρώνων μεταξύ των 3 hidden layers, όπως H1=256, H2=128, H3=32, επίσης H1=256, H2=64, H3=16, ή και H1=128, H2=32, H3=16 αλλά τα καλύτερα αποτελέσματα τα έδινε η αρχική διάταξη H1=128, H2=64, H3=32 του βασικού μας μοντέλου.

0.8.4 Base model - Learning rate

Για learning rate = 1e-5 έχουμε χαμηλό precision ενώ για learning rate = 1e-3 τείνει να κάνει overfit οπότε θα κρατήσουμε το αρχικό learning rate = 1e-4.

0.8.5 Base model - Optimizer

Πειραματίστηκα με αρκετούς optimizers πέρα από τον αρχικό SGD όπως: Adagrad, Adam, NAdam, Adamax, Nesterov, Momentum. Τα καλύτερα αποτελέσματα τα έδωσαν οι Adam, NAdam, Adamax. Τελικά θα ενσωματώσουμε στο μοντέλο μας τον optimazer Adamax καθώς αυτός έδωσε λίγο καλύτερα αποτελέσματα από τους προηγούμενους. Συγκεκριμένα:

Adamax	Loss	Accuracy	Precision	Recall	f1
Train Set	8.35386	0.84289	0.84212	0.84477	0.84107
Test Set	8.56920	0.84065	0.82683	0.86041	0.84193

0.8.6 Loss Function

Από τις διαφορετικές loss function που τέσταρα, όπως Binary Cross Entropy, HuberLoss η αρχική MSELoss έχει καλύτερα αποτελέσματα.

0.8.7 Activation Function

Δοκιμάζοντας τις ReLu, SELU, Tahn activations functions φαίνεται πως η ReLu λειτουργεί καλύτερα στο μοντέλο.

ReLu	Loss	Accuracy	Precision	Recall	f1
Train Set	7.19310	0.84309	0.84260	0.84449	0.84120
Test Set	7.48656	0.84109	0.81795	0.87520	0.84416

0.8.8 Dropout

Το dropout δεν είχε κάποια ιδιαίτερη επίδραση στην απόδοση του μοντέλου οπότε δεν το συμπεριέβαλα στο τελικό σκελετό.

0.9 Initialize the Train and Test Dataloader

0.9.1 Batch Size

Παρατηρούμε ότι είτε με batch size=64 είτε με batch size=32 φτάνουμε στα ίδια losses και accuracy ενώ άμα επιλέξουμε μεγαλύτερο batch size = 124 τότε θα έχουμε αρκετό loss. Οπότε κρατάμε το αρχικό μέγεθος ίσο με 64.

0.9.2 Number of Epoch

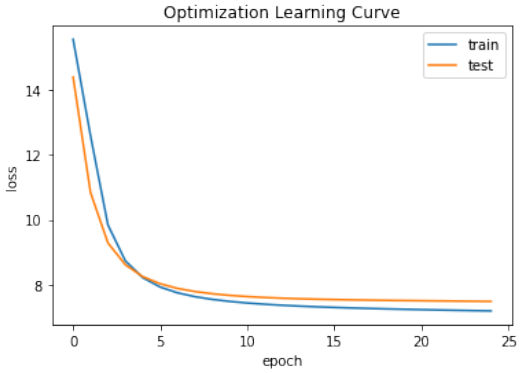
Μπορούμε να διακρίνουμε τρέχοντας το μοντέλο για περισσότερες από 25 epoch ότι αρχίζει να κάνει overfit καθώς οι καμπύλες των train και test set αρχίζου να απομακρύνονται.

0.10 Plots

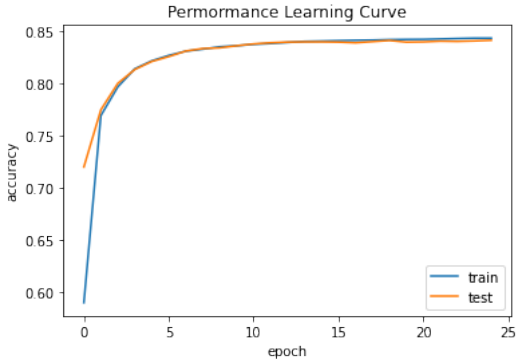
0.10.1 Comments

Παρατηρούμε ότι δεν παρουσιάζεται overfitting underfitting στις γραφικές καθώς οι καμπύλες βρίσκονται πολύ κοντά η μία στην άλλη.

0.10.2 Optimization Learning Curve



0.10.3 Performance Learning Curve



0.10.4 ROC curve