Τεχνητή Νοημοσύνη ΙΙ

Εργασία 3

REPORT - PROJECT DOCUMENTATION

Φουκανέλης Χρήστος-Γεώργιος, 1115201900204

Ιανουάριος 2022

Περιγραφή υλοποίησης

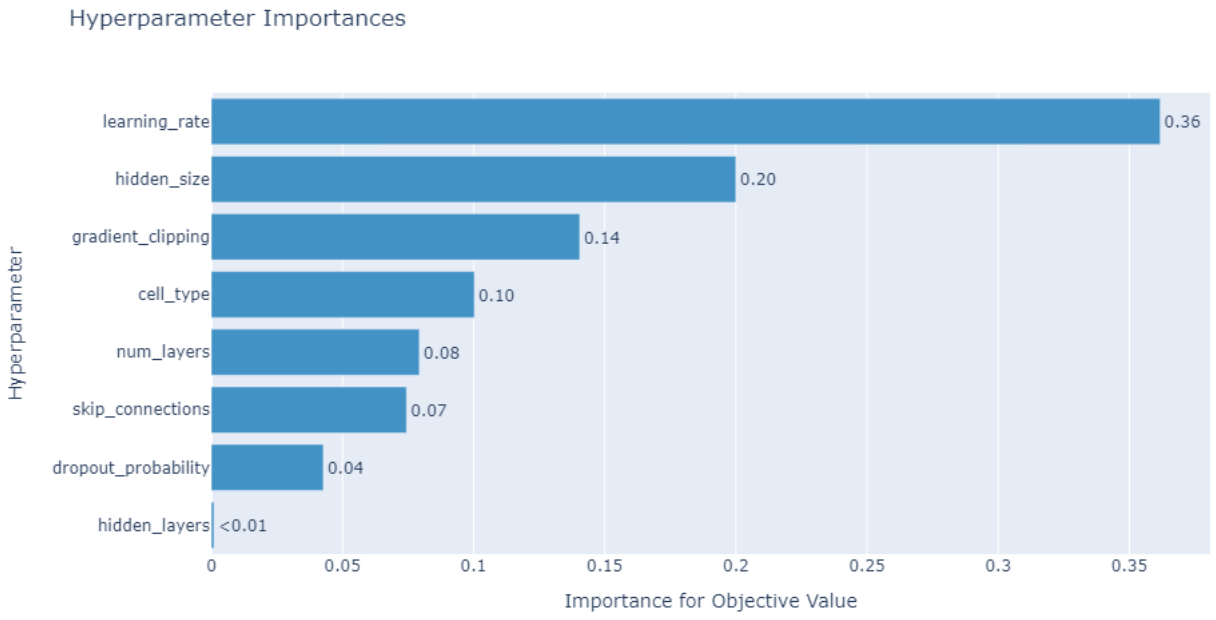
ΣΗΜΑΝΤΙΚΗ ΣΗΜΕΙΩΣΗ ΣΤΟ ΤΕΛΟΣ

Η εργασία υλοποιήθηκε με βάση τις οδηγίες της εκφώνησης και του φροντιστηρίου. Για την εύρεση των καλύτερων υπερπαραμέτρων των μοντέλων, χρησμιμοποιήθηκε η βιβλιοθήκη *optuna*. Έχουν υλοποιηθεί συναρτήσεις οι οποίες αρχικοποιούν το μοντέλο με τις υπερπαραμέτρους, το εκπαιδεύουν και το αξιολογούν με αυτόματο τρόπο. Ενδεικτικά παραθέτονται 2 παραδείγματα με διαφορετικές αρχιτεκτονικές δικτύου και επιλογές υπερπαραμέτρων. Για τα word embeddibngs χρησιμοποιήθηκαν vectors 200 διαστάσεων καθώς ήταν αρκετά αποδοτικά. Όσον αφορα τον καθαρισμό των δεδομένων έχουν εφαρμοστεί οι ίδιες τεχνικές που περιγράφονται στο report της 1ης εργασίας. Οι συναρτήσεις *objective, train model,train and evaluate, evaluate model* εχουν υλοποιηθεί με βάση το documentation της optuna και εκπαιδεύουν και αξιολογούν ένα μοντέλο με βάση της παραμέτρους που θα λάβουν οι οποίες έχουν παραχθεί κατά την εξερεύνηση του χώρου αναζήτης των υπερπαραμέτρων μεσω της create study. Η test model τεστάρει το μοντέλο πάνω σε ξένα δεδομένα και εκτυπώνει τις επιδόσεις του.

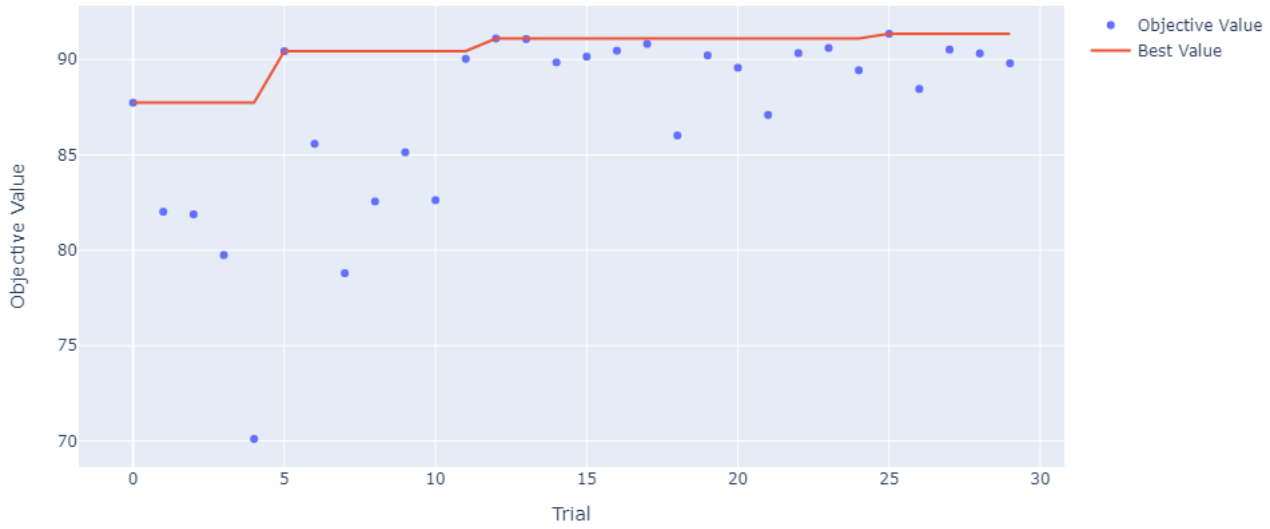
Υπερπαραμετροι και σχολιασμος

Η *study* της *optuna* διασχίζει ένα χώρο αναζήτησης με πολλές υπερπαραμέτρους. Ειδικότερα, δοκιμάστηκαν παραλλαγές για το *learning rate*, τον αριθμό των *stacked rnn's*, αν θα γίνει *skip connections* η οχι, το *dropout probability* στα ενδιάμεσα *layers*, τα *hidden layers*, το αν θα γίνει *gradient clipping*, τον τύπο του κελιού *rnn* αλλά και το *hidden size* των *rnn*. Δεν χρησιμοποιήθηκε *batch normalisation*.

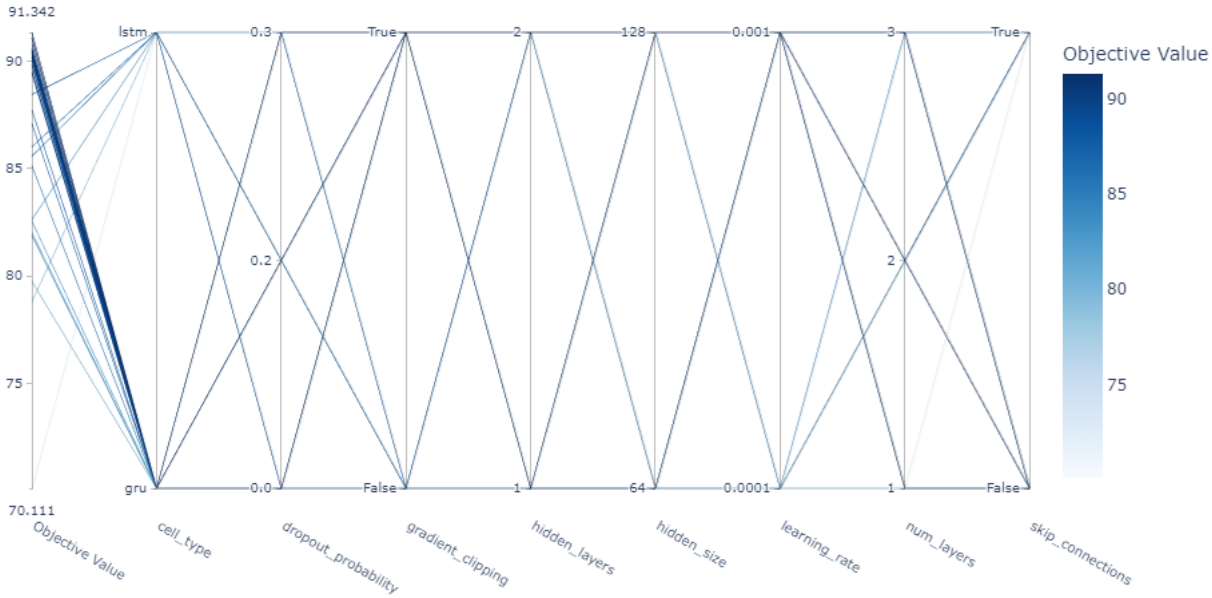
Η *optuna* επέστρεψε τις παραμέτρους με τις οποίες το μοντέλο είχε το περισσότερο *validation accuracy*. Πολλές είναι οι πιθανές παράμετροι καθώς διάφοροι συνδυασμοί δίνουν αποτέλεσμα πολύ κοντά σε αυτό του βέλτιστου όπως φαίνεται και στο optimization history της optuna. Παραθέτονται τα αντίστοιχα plots που καταδεικνείουν την σχέση μεταξύ των υπερπαραμέτρων και του αποτελέσματος. Αποτι φαίνεται, όλες οι αρχιτεκτονικές είχαν εξίσου καλές αποδόσεις και εν τέλει διαφορά έκανε το learning rate και το hidden size. Η lstm και η gru φαίνεται να έχουν παρόμοια απόσοση στους συνδυασμούς που έγιναν.

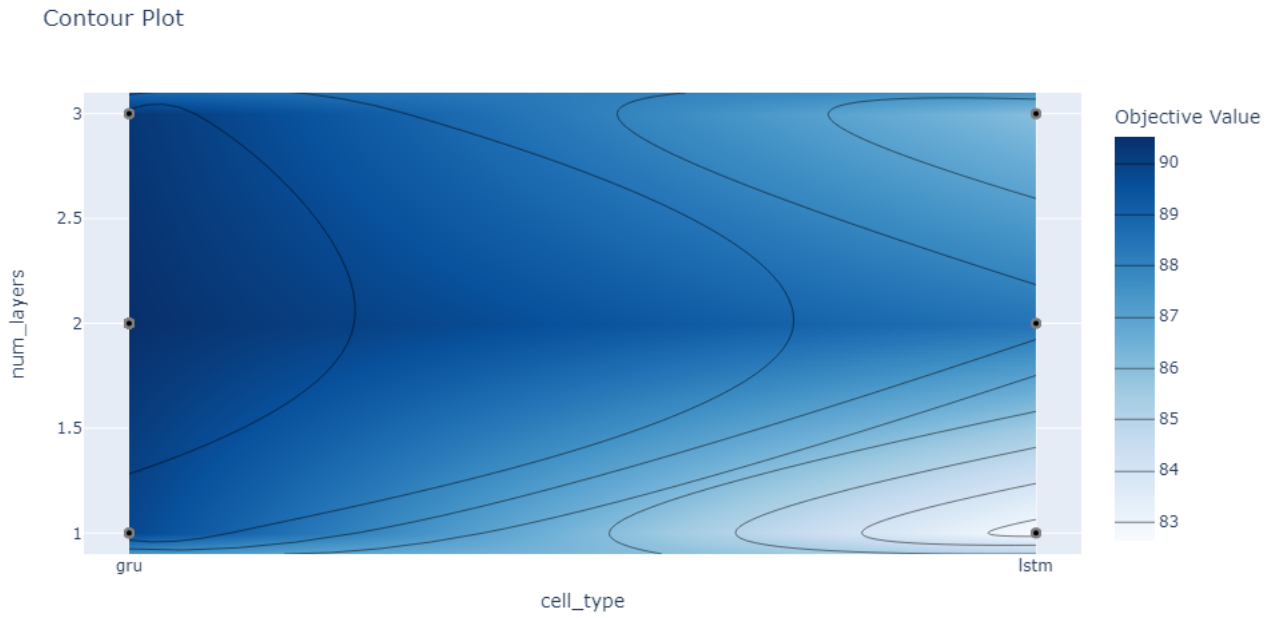


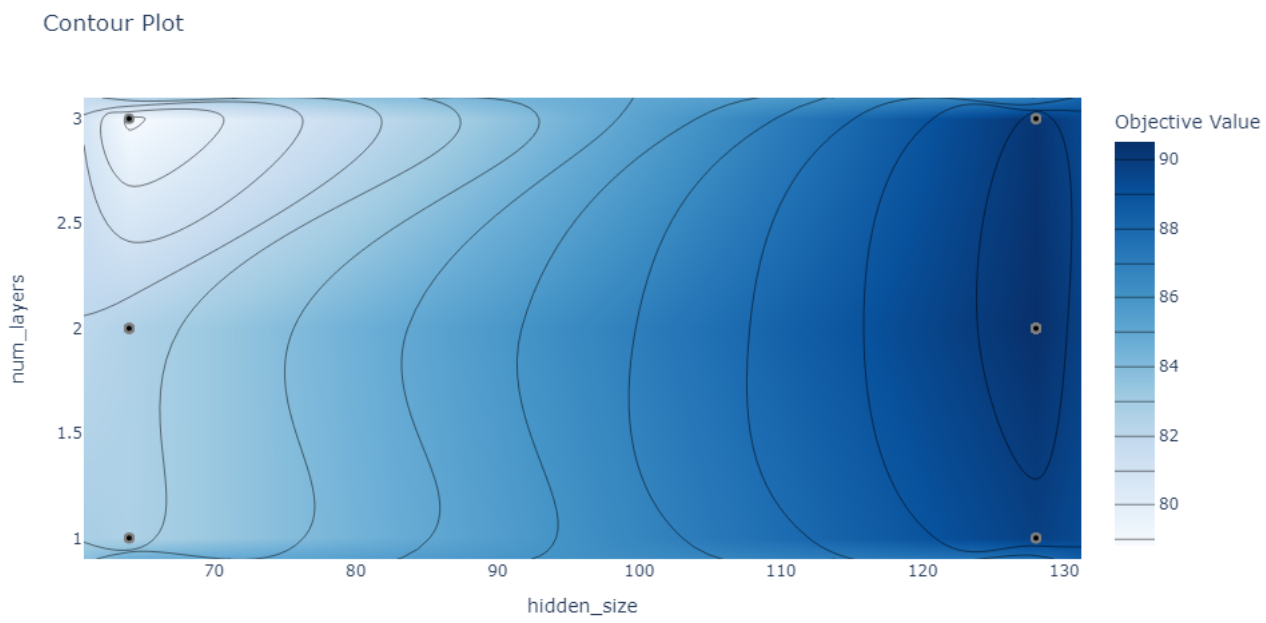
Optimization History Plot



Parallel Coordinate Plot

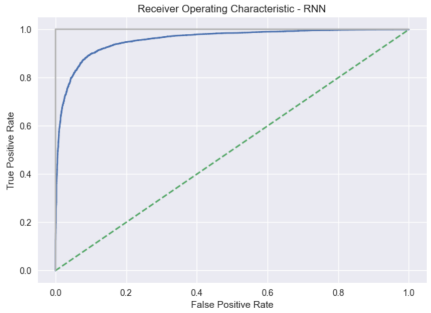



Contour Plot
num_layers
3
2.5
2
1.5
1
gru
lstm
cell_type
Objective Value
90
89
88
87
86
85
84
83


Contour Plot
num_layers
3
2.5
2
1.5
1
70
80
90
100
110
120
130
hidden_size
Objective Value
90
88
86
84
82
80

Ενδεικτικές μετρικές ενός μοντέλου


```
roc_auc_score for model:  0.9579897557411686
```



Συγκρίσεις μεταξύ μοντέλων:

LOGISTIC REGRESSION

```
Model Evaluation:
accurancy:  0.9027993779160186
precision:  0.9030501835088105
recall:  0.9027993779160186
f-measure:  0.9027827702847243
```

FEED-FORWARD NEURAL-NETWORK

```
Model confusion matrix:
 [[3684  808]
 [ 683 3827]]
--------------------------------------------
Model classification report:
              precision    recall  f1-score   support

         0.0       0.84      0.82      0.83      4492
         1.0       0.83      0.85      0.84      4510

    accuracy                           0.83      9002
   macro avg       0.83      0.83      0.83      9002
weighted avg       0.83      0.83      0.83      9002

--------------------------------------------
Model accuracy:  0.8343701399688958
--------------------------------------------
```

RNN

```
Model classification report:
              precision    recall  f1-score   support

         0.0       0.87      0.93      0.90      5627
         1.0       0.92      0.86      0.89      5625

    accuracy                           0.90     11252
   macro avg       0.90      0.90      0.90     11252
weighted avg       0.90      0.90      0.90     11252
```

Όπως φαίνεται, το RNN και το logistic regression εχουν παρομοια απόδοση γύρω στο 0,9 με αναμενόμενα το feed-forward να είναι πιο κάτω. Με μεγαλύτερη διάσταση embeddings και περισσότερες εποχές το RNN θα είχε ακόμα καλύτερα αποτελέσματα.

Σημειώσεις - Παρατηρήσεις:

ΣΗΜΑΝΤΙΚΗ ΣΗΜΕΙΩΣΗ

Όπως αναφέρθηκε και στο piazza στην ερώτηση που υπέβαλλα (@88), για κάποιο λόγο ο οποίος δεν είναι άμεσα ξεκάθαρος και απαιτεί πλήρες refactor του κώδικα για να λυθεί, δεν μπορούσα να αποθηκεύσω το μοντέλο με torch.jit οπότε χρησιμοποίησα torch.save. Ενδεικτικά δοκιμάστηκαν 2 διαφορετικές αρχιτεκοτινικές και στο ξεχωριστό notebook φορτώνεται ένα απο αυτά τα μοντέλα. Έχουν επιλεχθεί οι 8 εποχές ώστε να μην διαρκέσει υπερβολικό χρόνο η εκπαίδευση αν και με μικρή αύξηση ενδέχεται η απόδοση του μοντέλου να αυξηθεί. Επιπλέον έγινε απόπειρα υλοποίησης του attention αλλά δεν έγινε integrate στο project καθώς ηταν μέρος του προβλήματος αποθήκευσης του μοντέλου. Όπως φαίνεται, το μοντελο πετυχαίνει μια καλή ακρίβεια της τάξης του 0,9. Στο παραδοτέο συμπεριλαμβάνονται: Το ipynb αρχείο με τις δοκιμές, το test notebook, το readme και το .tex αρχείο (ενδέχεται να υπαρχουν errors καθώς έγινε compile σε online editor), το εκπαιδευμένο μοντέλο και τα διαφορετικά csv που παράχθηκαν απο το split. ***__To test notebook__ είναι γραμμένο έτσι ώστε αρκεί να εισαχθεί το path για ένα csv αρχείο, και θα εκτελεστεί η αξιολόγηση του μοντέλου. Αρκεί να εισαχθεί το path του αρχείου στη μεταβλητή *TEST FILE PATH*. Το notebook κατεβάζει τα embeddings επιτόπου οπότε αρκεί ενα run all για να τρέξουν όλα τα cells και να εκτελεστεί το pipeline***