2η Εργασία

Υλοποίηση αλγορίθμου με ***Logistic Regression.***

Αρχικά, θα πρέπει να συμπεριληφθεί στον φάκελο της εργασίας ολόκληρο το αρχείο aclImdb με τις κριτικές.

Παρατηρήσεις σχετικά με το αρχείο aclImdb :

- Το imdb.vocab περιέχει όλες τις λέξεις που υπάρχουν στα δεδομένα εκπαίδευσης, σε φθίνουσα σειρά ως προς την συχνότητα που εμφανίζεται η καθε λέξη στα δεδομένα εκπαίδευσης (train data)
- Θα χρησιμοποιήσουμε τα labeledBow.feat αρχεία των train και test αρχείων για να αποθηκεύσουμε-επεξεργαστούμε τα δεδομένα των κριτικών.

def read_vocabulary_aclImdb(file_name): Δέχεται ως όρισμα το αρχείο imdb.vocab. Σε κάθε λέξη δίνουμε μία μοναδική τιμή int ID, ξεκινώντας από την πιο συχνή λέξη και αντιστοιχώντας την τιμή ID = 0 και αυξάνουμε το ID κατά 1 για κάθε επόμενη λέξη του vocabulary. (Με τον ίδιο τρόπο που χρησιμοποιούνται και στα αρχεία labeledBow.feat) Επιστρέφει τον πίνακα vocabulary περιέχει όλα τα IDs.

def read_dataset_aclImdb(file_name): Δέχεται ως όρισμα το αρχείο labeledBow.feat (είτε για τα train data είτε για τα test). Η συνάρτηση επιστρέφει 2 πίνακες με τις θετικές και αρνητικές κριτικές του κάθε dataset. Κάθε πίνακας περιλαμβάνει για κάθε κριτική, μία λίστα με τα ID των λέξεων του vocabulary που υπάρχουν στην συγκεκριμένη κριτική. Π.χ: [[3, 4, 5, 15, 58, 69, 566, 599, 621...],
[2, 3, 5, 9, 15, 98, 124 ,126, 199...]
...
]

def convert_data: Δέχεται ως όρισμα έναν πίνακα (πρακτικά τα θετικά/αρνητικά παραδείγματα που θα φτιαχνούν από την παραπάνω συνάρτηση) και την τιμή c που είναι 0 ή 1 και δείχνει την κατηγορία των παραδειγμάτων.
Θα επιστρέψει ένα διάνυσμα x όπου για κάθε παράδειγμα, θα έχει μία λίστα. Μέσα στη λίστα ενός παραδείγματος θα έχει σε κάθε θέση τιμή 0/1 ανάλογα με το αν υπάρχουν στο παράδειγμα αυτό, οι λέξεις του vocabulary που θα υπολογίσουμε μέσω της παρακάτων κλασης Information Gain. Τέλος, επιστρέφει ένα διάνυσμα/λίστα y όπου δείχνει την κατηγορία των παραδειγμάτων του x (η κατηγορία σε κάθε κλήση είναι η c-όρισμα της μεθόδου-)

class Information_Gain: Χρησιμοποιώντας την κλάση αυτή, θα βρούμε τις m λέξεις με το υψηλότερο Information Gain για να εκπαιδεύσουμε τον αλγόριθμο Logistic Regression.

- *def __init__():* δέχεται τον πίνακα vocabulary με τα ID και τους πίνακες με τα positive_examples και negative_examples του training dataset. Αρχικοποιούμε την εντροπία H_C = 1, καθώς έχουμε ίσο αριθμό θετικών και αρνητικών παραδειγμάτων εκπαίδευσης στο dataset. (Προφανώς, για άλλο dataset μπορούμε να αλλάξουμε την τιμή της H_C)
- *def calculate_IG():* δέχεται ως όρισμα την int μεταβλητή n που δείχνει το πλήθος των λέξεων στις οποίες θα υπολογιστεί το Information Gain. (Θα είναι οι πρώτες n λέξεις του πίνακα vocabulary -οι n πιο συχνές λέξεις-). Υπολογίζει το IG για κάθε λέξη και αποθηκέυεται στον πίνακα features μία λίστα [word, IG(word)] για την λέξη αυτή, όπου word το ID της λέξης και IG(word) το information gain. Ταξινομούμε τον πίνακα features σε φθίνουσα σειρά βάσει του information gain($2^{ου}$ στοιχείου κάθε λίστας του πίνακα) με χρήση της βοηθητικής συνάρτησης myFunc().
- *def IG():* Παίρνει ως όρισμα ένα ID(μία λέξη) και υπολογίζει και επιστρέφει το information gain για τη λέξη αυτή.
- *def get_m_feature():* Επιστρέφει έναν πίνακα με τα ID των m πρώτων λέξεων, του πίνακα features (τις m λέξεις με το υψηλότερο information gain).

Αρχικοποίηση δεδομένων:

1. Διαβάζουμε και αποθηκεύουμε στον πίνακα vocabulary το λεξιλόγιο του aclImdb με χρήση της read_vocabulary_aclImdb().
2. Διαβάζουμε και αποθηκεύουμε στους πίνακες positive_examples, negative_examples τα θετικά και αρνητικά παραδείγματα του train, όπως περιγράψαμε παραπάνω με χρήση της read_dataset_aclImdb().
3. Διαβάζουμε και αποθηκεύουμε σε πίνακες τα παραδείγματα του test dataset, με χρήση της read_dataset_aclImdb().
4. Υπολογίζουμε το IG για τις n = 5000 συχνότερες λέξεις.
5. Επιλέγουμε τις m = 1000 λέξεις με το υψηλότερο IG και τις αποθηκεύουμε στον πίνακα vocabulary.
6. Αρχικοποιούμε μέσω της convert_data τους πίνακες x_train, y_train, x_den, y_dev, x_test, y_test. (Το 20% των train data θα χρησιμοποιηθεί για development)

class LogisticRegression:

__init__: Αρχικοποιεί τα βάρη σε None και τον πίνακα lr_rates που δείχνει τα learning rates που θα χρησιμοποιήσουμε κατά την εκπαίδευση.
Έχουμε τις private βοηθητικές μεθόδους __sigmoid, __gradient_descent:

- **__sigmoid**: Υλοποίηση της σιγμοειδούς συνάρτησης που επιστρέφει την πιθανότητα μία συγκεκριμένη κριτική με διάνυσμα $\vec{x}$ και βάρη $\vec{w}$, να είναι θετική.
- **__gradient_descent**: Μεγιστοποίηση πιθανοφάνειας με στοχαστική ανάβαση κλίσης και για την ακρίβεια μεγιστοποιούμε το $l(\vec{w}) - \lambda^*\|\vec{w}\|^2 = l(\vec{w}) - \lambda^*\sum_{l=0}^{n} w_l^2$.
  Συγκεκριμένα, ο κανόνας ενημέρωσης βαρών είναι:
  $\vec{w} \leftarrow \vec{w} + \eta^*\nabla\vec{w}(l(\vec{w}) - \lambda^*\sum_{l=0}^{n} w_l^2)$.
- **__calculate_w:** Δέχεται ως όρισμα παραδείγματα training με τις σωστές κατηγορίες, τον όρο κανονικοποίησης n_t και το learning rate lr.
  Υπολογίζει τα w βάση των παραπάνω παραμέτρων.

Τέλος, έχουμε τις public μεθόδους:

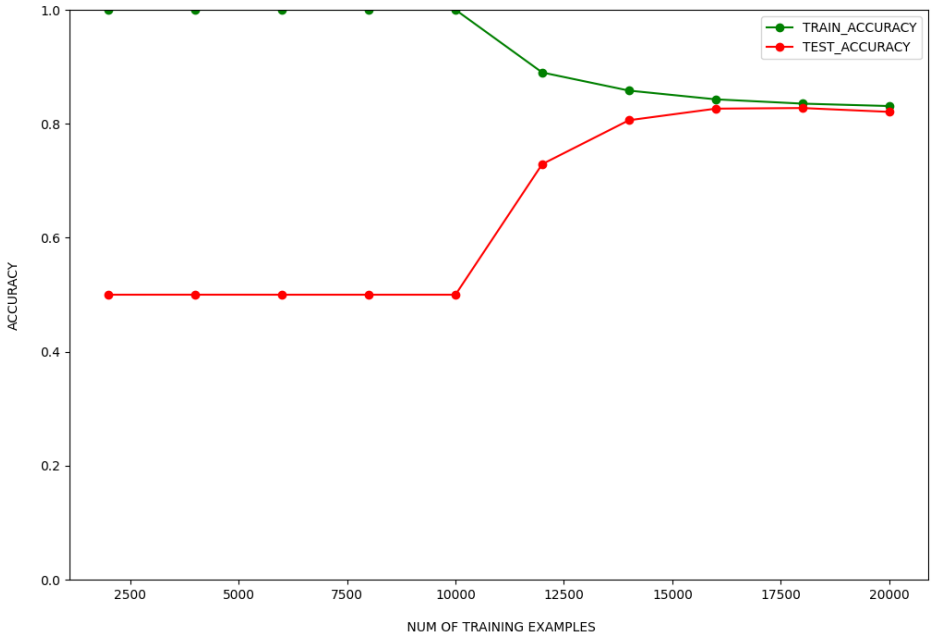
- **fit_hyperparameters**: Δέχεται ως όρισμα τα train και dev data μαζί με τις σωστές κατηγορίες τους(σε μορφή πίνακα κάθε όρισμα). Σκοπός είναι να βρεθεί ο κατάλληλος όρος κανονικοποίησης όπου μεγιστοποιούμε το development accuracy.
  Εξετάζουμε τις τιμές 0.05, 0.1, 0.15, 0.2.
  Για κάθε όρο κανονικοποίησης εκπαιδεύουμε τον αλγόριθμο ως εξής:
  Για κάθε learning rate(πίνακα που έχει αρχικοποιηθεί) θα «κατευθύνουμε» τα βάρη w το πολύ 5 εποχές.
  Στο τέλος θα βρούμε το development accuracy κάθε όρου και θα επιέξουμε τον όρο κανονικοποίησης με το μεγαλύτερο dev_accuracy.
- **fit:** Δέχεται ως όρισμα τα train data και τις σωστές κατηγορίες για κάθε παράδειγμα. Πρατικά, εκπαιδεύει τον αλγόριθμο για την εύρεση των βαρών w, με τη διαφορά ότι χρησιμοποιεί τις υπερπαραμέτρους που έχουν αρχικοποιηθεί μετά την κήση της **fit_hyperparameters** (δηλαδή με τον όρο κανονικοποίησης λ)**.**
- **accurate:** Δέχεται ως όρισμα ένα σετ παραδειγμάτων (train/dev/test), τις σωστές κατηγορίες τους και τιμή True ή False για την παράμετρο All.
  Εάν η All είναι false επιστρέφει το accuracy για το δεδομένο σετ παραδειγμάτων, διαφορετικά επιστρέφει το accuracy, precision, recall και F1. Τα precision, recall και F1 υπολογίζονται για την κατηγορία των θετικών παραδειγμάτων – κριτικών.
- **predict:** Δέχεται ως όρισμα ένα dataset παραδειγμάτων και επιστρέφει έναν πίνακα με την κατηγοριοποίηση που έκανε ο αλγόριθμός μας για κάθες παράδειγμα-κριτική, σύμφωνα με τα βάρη w που έχουν υπολογιστεί ήδη.

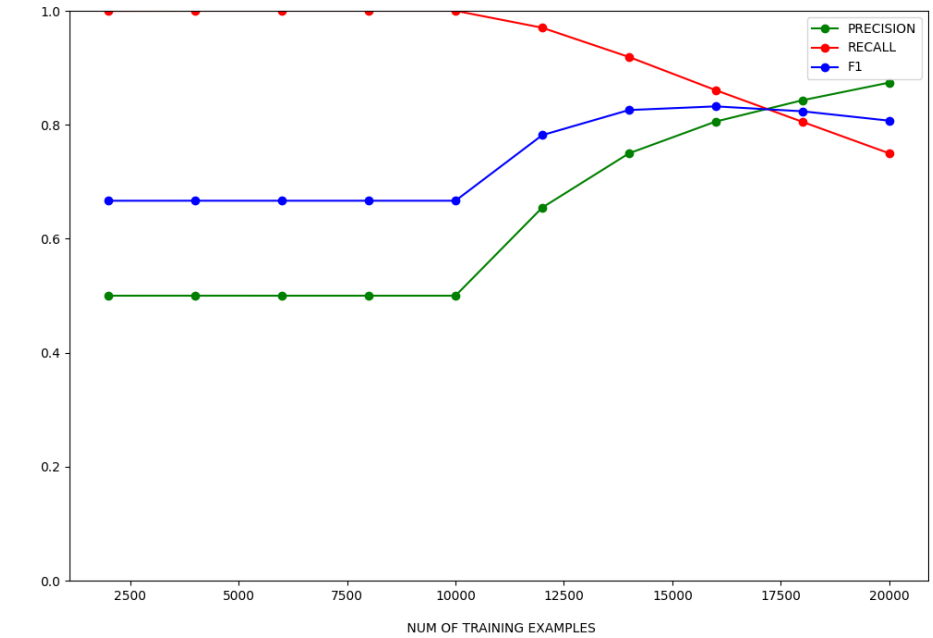
Υπερπαράμετροι : όρος κανονικοποίησης = 0.1

```
print(lr.best_normalization_term)
```

```
0.1
```

Learning Curves





Tables

Training examples	Train Accuracy	Dev Accuracy	Test Accuracy
2000	1.0	0.5	0.5
4000	1.0	0.5	0.50004
6000	1.0	0.5	0.5
8000	1.0	0.5	0.5
10000	1.0	0.5	0.5
12000	0.89	0.728	0.72916
14000	0.8582142857142857	0.7984	0.8062
16000	0.842875	0.827	0.82652
18000	0.8353333333333334	0.8318	0.8276
20000	0.8311	0.8242	0.82088

Training examples	Precision	Recall	F1
2000	0.5	0.99984	0.6666311073181139
4000	0.500020000800032	1.0	0.6666844449185311
6000	0.5	1.0	0.6666666666666666
8000	0.5	1.0	0.6666666666666666
10000	0.5	1.0	0.6666666666666666
12000	0.6545954989475956	0.97032	0.7817847819781495
14000	0.7498531235720347	0.91896	0.8258384557316942
16000	0.8056158742044178	0.86072	0.832256816863276
18000	0.8430222817892444	0.80512	0.8236353220394467
20000	0.8739511467462241	0.74992	0.8071988288986481

Πίνακες σύγκρισης με το αλγόριθμο LogisticRegression της sklearn

```
# sklearn.LogisticRegression report
print(classification_report(y_test, log.predict(x_test)))
```

```
              precision    recall  f1-score   support

           0       0.88      0.87      0.87     12500
           1       0.87      0.88      0.87     12500

    accuracy                           0.87     25000
   macro avg       0.87      0.87      0.87     25000
weighted avg       0.87      0.87      0.87     25000
```

```
# our report
print(classification_report(y_test, lr.predict(x_test)))
```

```
              precision    recall  f1-score   support

           0       0.78      0.89      0.83     12500
           1       0.87      0.75      0.81     12500

    accuracy                           0.82     25000
   macro avg       0.83      0.82      0.82     25000
weighted avg       0.83      0.82      0.82     25000
```