Τμήμα Πληροφορικής και Τηλεματικής
Χαροκόπειο Πανεπιστήμιο
ΕΠ34 Μηχανική Μάθηση και Εφαρμογές

1η Άσκηση: Λογιστική παλινδρόμηση

Έκδοση 1.0

Διδάσκων: Χρήστος Δίου

1 Εισαγωγή

Στην άσκηση αυτή θα υλοποιήσουμε και θα αξιολογήσουμε μία μέθοδο γραμμικής ταξινόμησης που βασίζεται στη λογιστική παλινδρόμηση. Η άσκηση σας καθοδηγεί βήμα-βήμα στην υλοποίηση.

Παραδοτέα

Για την υποβολή της εργασίας, θα χρειαστεί να υποβληθούν, σε ένα zip file

- Ένα αρχείο `logistic_regression.py` καθώς και 3 αρχεία `experiment1.py`, `experiment2.py` και `experiment3.py` με τον κώδικα σε γλώσσα python, με επαρκή σχόλια. Ο κώδικας θα πρέπει να μπορεί να εκτελεστεί σε περιβάλλον python3 με πρόσφατες εκδόσεις των βιβλιοθηκών `numpy` και `scikit-learn`
- Μια συνοπτική αναφορά που να περιγράφει τα αποτελέσματα της εκτέλεσης του κώδικά σας καθώς και απαντήσεις στα ερωτήματα της εργασίας. Η αναφορά μπορεί να περιέχει και τυχόν λεπτομέρειες για τον τρόπο εκτέλεσης του κώδικα, αν το κρίνετε απαραίτητο.

Την άσκηση θα πρέπει να την υλοποιήσετε ατομικά.

2 Υλοποίηση μοντέλου λογιστικής παλινδρόμησης

Δημιουργήστε το αρχείο `logistic_regression.py` το οποίο θα υλοποιεί την κλάση

```
LogisticRegressionEP34
```

Η κλάση αυτή υλοποιεί ένα μοντέλο λογιστικής παλινδρόμησης. Για την υλοποίηση πρέπει να χρησιμοποιήσετε τη βιβλιοθήκη `numpy` και `matplotlib` της python και *καμία άλλη*.

Σημείωση: Προσπαθήστε όσο μπορείτε να υλοποιήσετε τις παρακάτω μεθόδους χρησιμοποιώντας πράξεις μεταξύ πινάκων και διανυσμάτων. Αυτό θα επιταχύνει πολύ σημαντικά το χρόνο εκτέλεσης του κώδικά σας, καθώς θα αξιοποιεί τη βελτιστοποιημένη υλοποίηση της `numpy` για πράξεις γραμμικής άλγεβρας.

2.1 Κλάση και ιδιότητες

Η `LogisticRegressionEP34` έχει τις εξής ιδιότητες (attributes)

- `w` : Διάνυσμα numpy το οποίο αντιστοιχεί στα βάρη, $\mathbf{w}$, του μοντέλου
- `b` : Αριθμός που αντιστοιχεί στον όρο μεροληψίας, b, του μοντέλου

- `lr` : Ρυθμός εκμάθησης για την εκπαίδευση του μοντέλου

Θα χρειαστεί να προσθέσετε και άλλες ιδιότητες στο μοντέλο σας, όπως το πλήθος των δεδομένων N με τα οποία εκπαιδεύτηκε, ή η διάσταση p του χώρου των χαρακτηριστικών, όπως και προσωρινές μεταβλητές όπως οι παράγωγοι των $\mathbf{w}$ και b σε κάθε βήμα βελτιστοποίησης, καθώς και η πρόβλεψη $p(1|\mathbf{x})$ για τα δεδομένα εισόδου $\mathbf{x}$ κατά την εκπαίδευση (δείτε τις επόμενες ενότητες). Οι ιδιότητες της κλάσης μπορούν να αρχικοποιούνται με **`None`** (στην `__init__()`), εκτός από το ρυθμό εκμάθησης. Αυτός πρέπει να μπορεί να δοθεί ως είσοδος στην `__init__()` , με προεπιλεγμένη τιμή 10^{-2}.

Οι μέθοδοι της `LogisticRegressionEP34` περιγράφονται στα ακόλουθα.

2.2 `init_parameters(self)`

Η μέθοδος αρχικοποιεί τις παραμέτρους $\mathbf{w}$ και b με δειγματοληψία από Γκαουσιανή κατανομή με μέση τιμή $\mu = 0$ και τυπική απόκλιση $\sigma = 0.1$. Αυτό μπορείτε να το πετύχετε με τη χρήση της συνάρτησης `random.randn()` .

2.3 `forward(self, X)`

Δέχεται ως είσοδο έναν $N \times p$ πίνακα $\mathbf{X}$ κάθε γραμμή του οποίου αντιστοιχεί σε ένα δείγμα δεδομένων και υπολογίζει την τιμή

$$p_{\text{model}}(1|\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x}+b)}} \tag{1}$$

για κάθε δείγμα. Ένας τρόπος να το κάνετε αυτό ταυτόχρονα για όλες τις γραμμές του πίνακα είναι να υπολογίσετε ως όρισμα στη λογιστική συνάρτηση το διάνυσμα $\mathbf{X}\mathbf{w} + b\mathbb{1}_{N\times 1}$ (ή `X@self.w+self.b` , στην python). Η συνάρτηση επίσης αποθηκεύει το αποτέλεσμα του υπολογισμού σε μία μεταβλητή της κλάσης (πχ την `self.f`).

2.4 `predict(self, X)`

Κάνει ότι και η `forward` , χωρίς να αποθηκεύει το αποτέλεσμα στην `self.f` .

2.5 `loss(self, X, y)`

Υπολογίζει την τιμή της συνάρτησης απώλειας, δηλ. τη δυαδική διεντροπία του μοντέλου για τα δεδομένα `X, y` .

2.6 `backward(self, X, y)`

Υπολογίζει την παράγωγο της απώλειας ως προς κάθε παράμετρο, δηλ. το διάνυσμα $\nabla_{\boldsymbol{\theta}}\mathcal{L}$. Για τον υπολογισμό αυτό χρησιμοποιήστε ως δεδομένο ότι για τα w_i,

$$\frac{\partial \mathcal{L}}{\partial w_i} = -\frac{1}{N}\sum_{n=0}^{N}(y^{(n)} - p_{\text{model}}(1|\mathbf{x}^{(n)}))x_i^{(n)} \tag{2}$$

ενώ για το b

$$\frac{\partial \mathcal{L}}{\partial b} = -\frac{1}{N}\sum_{n=0}^{N}(y^{(n)} - p_{\text{model}}(1|\mathbf{x}^{(n)})) \tag{3}$$

Προσπαθήστε να κάνετε τον υπολογισμό αυτό χωρίς τη χρήση δομής επανάληψης (**`for`**). Οι υπολογισμένες παράγωγοι να αποθηκευτούν σε ιδιότητες `self.l_grad_w` και `self.l_grad_b` .

2.7 `step()`

Η συνάρτηση εκτελεί ένα βήμα του αλγόριθμου Gradient Descent, δηλ.

$$w_i \leftarrow w_i - \eta \frac{\partial \mathcal{L}}{\partial w_i}$$

και αντίστοιχα για το b.

2.8 `fit()`

Η πλήρης δήλωση της συνάρτησης είναι

```
fit(self, X, y, iterations=10000, batch_size=None,
    show_step=1000, show_line=False)
```

Η `fit` δέχεται στην είσοδο έναν $N \times p$ πίνακα σχεδιασμού, `X`, κάθε γραμμή του οποίου αντιστοιχεί σε ένα διάνυσμα χαρακτηριστικών του συνόλου εκπαίδευσης, και ένα $N \times 1$ διάνυσμα τιμών εξόδου `y`. Ο πίνακας και τα διανύσματα να είναι numpy arrays. Η συνάρτηση πρέπει να κάνει τα ακόλουθα:

- Να επιβεβαιώνει ότι τα `X` και `y` είναι numpy arrays και ότι οι μεταξύ τους διαστάσεις είναι συμβατές. Σε διαφορετική περίπτωση πρέπει να ενεργοποιεί `ValueError` exception (ή κάποιο άλλο αντίστοιχο)
- Να αρχικοποιεί τις παραμέτρους του μοντέλου καλώντας τη συνάρτηση `init_parameters()`
- Να τοποθετεί τα δείγματα (γραμμές) στον `X` σε τυχαία σειρά

Έπειτα πρέπει να ξεκινάει την επανάληψη εκπαίδευσης, όπου

- Επιλέγονται τα επόμενα `batch_size` δείγματα. Αν έχουμε φτάσει στο τέλος του πίνακα ξεκινάμε πάλι από την αρχή (μπορεί το τελευταίο batch να έχει λιγότερα από `batch_size` δείγματα). Αν το `batch_size` είναι `None` τότε σε κάθε επανάληψη επιλέγονται όλα τα δείγματα.
- Καλούνται διαδοχικά η `forward()` (υπολογίζει την `self.f`), η `backward()` (υπολογίζει τις παραγώγους) και η `step()` (εκτελεί ένα βήμα του gradient descent).
- Αν έχουν εκτελεστεί `show_step` βήματα, τότε καλεί τη `loss` για όλα τα δείγματα του συνόλου εκπαίδευσης και εκτυπώνει την τρέχουσα επανάληψη (πχ `1000`) καθώς και την τρέχουσα τιμή του `loss`. Αν η παράμετρος `show_line` είναι `True`, τότε επίσης καλεί την `show_line()` (δείτε παρακάτω)

Η διαδικασία αυτή επαναλαμβάνεται για `iterations` επαναλήψεις. Η `fit` δεν επιστρέφει κάτι.

2.9 Βοηθητική μέθοδος: `show_line(self, X, y)`

Η μέθοδος αυτή είναι χρήσιμη για αποσφαλμάτωση. Όταν το dataset έχει δύο διαστάσεις, τότε εμφανίζει ένα διάγραμμα όπου (α) εμφανίζονται τα σημεία των δύο κλάσεων και (β) εμφανίζεται η γραμμή που αντιστοιχεί στο μοντέλο. Ο κώδικάς της φαίνεται παρακάτω.

```
def show_line(self, X: np.ndarray, y: np.ndarray) -> None:
    """
    Plot data points for two classes, as well as the line
    corresponding to the model.
    """
    if (X.shape[1] != 2):
        print("Not plotting: Data is not 2-dimensional")
        return
    idx0 = (y == 0)
```

```
    idx1 = (y == 1)
    X0 = X[idx0, :2]
    X1 = X[idx1, :2]
    plt.plot(X0[:, 0], X0[:, 1], 'gx')
    plt.plot(X1[:, 0], X1[:, 1], 'ro')
    min_x = np.min(X, axis=0)
    max_x = np.max(X, axis=0)
    xline = np.arange(min_x[0], max_x[0], (max_x[0] - min_x[0]) / 100)
    yline = (self.w[0]*xline + self.b) / (-self.w[1])
    plt.plot(xline, yline, 'b')
    plt.show()
```

Το αρχείο `logistic_regression.py` και την κλάση που περιέχει θα το χρησιμοποιήσετε στα επόμενα πειράματα.

3 Πείραμα 1: Συνθετικά δεδομένα

Από αυτό το σημείο και έπειτα μπορείτε να χρησιμοποιήσετε και ορισμένες συναρτήσεις και κλάσεις της βιβλιοθήκης `scikit-learn`. Χρησμοποιήστε το αρχείο `generate_dataset.py` που σας δίνεται μαζί με την εκφώνηση της εργασίας. Καλέστε την `generate_binary_problem` για να παράξετε $N = 1000$ σημεία με κέντρα στο $(0, 0)$ και $(8, 8)$, δηλ. κέντρα που δίνονται από τον πίνακα

$$\mathbf{C} = \begin{bmatrix} 0 & 8 \\ 0 & 8 \end{bmatrix} \tag{4}$$

Χωρίστε το σύνολο δεδομένων σε 70% σύνολο εκπαίδευσης και 30% σύνολο δοκιμής. Αρχικοποιήστε ένα αντικείμενο `LogisticRegressionEP34` και καλέστε την `fit` για τα δεδομένα εκπαίδευσης με παραμέτρους `show_line = True` και `batch_size = None`. Τι παρατηρείτε για τη γραμμή και για την τιμή της συνάρτησης απώλειας, καθώς προχωράει η εκπαίδευση του μοντέλου; Υπολογίστε και την ευστοχία του μοντέλου μετά την εκπαίδευση.

Δοκιμάστε και γράψτε τις παρατηρήσεις σας και για άλλα, διαφορετικά κέντρα των κλάσεων, που αντιστοιχούν σε πιο δύσκολα διαχωρίσιμα προβλήματα.

4 Πείραμα 2: Breast cancer dataset

Φορτώστε το σύνολο δεδομένων Breast Cancer του scikit-learn (δείτε το σχετικό notebook). Εφαρμόστε τα ακόλουθα βήματα για 20 επαναλήψεις:

- Χωρίστε το σύνολο δεδομένων σε 70% δεδομένα εκπαίδευσης και 30% δοκιμής.
- Κανονικοποιήστε τα δεδομένα στο $[0, 1]$
- Εκπαιδεύστε ένα μοντέλο λογιστικής παλινδρόμησης με ρυθμό εκμάθησης 0.01 `batch_size = 64`
- Αξιολογήστε την ευστοχία στο σύνολο δοκιμής

Υπολογίστε και σημειώστε το μέσο όρο και την τυπική απόκλιση της ευστοχίας του μοντέλου για τις 20 επαναλήψεις. Πόσο χρόνο κάνει να τρέξει αυτό το πειράματα στον υπολογιστή σας; Αναφέρετε τον επεξεργαστή, τη μνήμη του υπολογιστή σας, καθώς και το χρόνο εκτέλεσης σε δευτερόλεπτα.

5 Πείραμα 3: Breast cancer dataset (scikit-learn)

Κάντε ακριβώς ότι κάνατε στο προηγούμενο πείραμα, αλλά χρησιμοποιώντας την κλάση `LogisticRegression` του `scikit-learn`, με τις προεπιλεγμένες παραμέτρους. Αναφέρετε και πάλι το χρόνο εκτέλεσης. Πως συγκρίνονται τα δύο μοντέλα, σε ότι αφορά την επίδοση και την αποτελεσματικότητά τους;