Εργασία Μηχανικής Μάθησης 2022-2023

Τσίρμπας Δημήτρης p3190205

22nd January 2023

Όλα τα ερωτήματα της εργασίας έχουν υλοποιηθεί.

1 Δομή Εργασίας

Η εργασία χωρίζεται σε 10 αρχεία πηγαίου κώδικα python:

- `load_mnist.py` το οποίο εκτελεί την φόρτωση και προ-επεξεργασία των δεδομένων
- `logistic_regression.py` περιέχει το μοντέλο λογιστικής παλινδρόμησης
- `run_logistic.py` περιέχει τις εντολές για την δημιουργία των γραφημάτων και αποτελεσμάτων του μέρους Β
- `mlp.py` περιέχει το μοντέλο απλού νευρωνικού δικτύου
- `run_mlp.py` περιέχει τις εντολές για την δημιουργία των γραφημάτων και αποτελεσμάτων του μέρους Γ (εκτός από το ερώτημα Ι)
- `gradcheck.py` το οποίο ελέγχει την μέθοδο backpropagation του ερωτήματος Ζ
- `sgd.py` περιέχει το μοντέλο του στοχαστικού νευρωνικού δικτύου
- `run_sgd.py` περιέχει τις εντολές για την δημιουργία των γραφημάτων και αποτελεσμάτων του ερωτήματος Ι

Περιέχονται επίσης τα αρχεία `common.py`, που περιέχει κοινές συναρτήσεις για τα παραπάνω αρχεία, και `test_load_mnist.py` το οποίο συμβάλλει στην επικύρωση των δεδομένων μας.

Η πλήρη τεκμηρίωση του κώδικα και της υλοποίησης μπορεί να βρεθεί στα παραπάνω αρχεία με την μορφή pydoc και σχολίων.

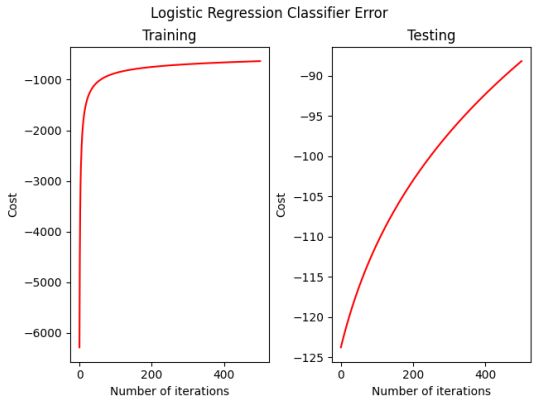

Figure 1: Τα αποτελέσματα της εκπαίδευσης και του ελέγχου στον ταξινομητή μας. Αριστερά: Το κόστος εκπαίδευσης ως συνάρτηση των επαναλήψεων του αλγορίθμου gradient ascent. Δεξία: Το αντίστοιχο κόστος ελέγχου. Υπενθυμίζουμε ότι οι κλίμακες των γραφημάτων δεν είναι ίσες, καθώς ο ήδη εκπαιδευμένος ταξινομητής αρχίζει με πολύ μικρότερο κόστος.

2 Μέρος Β - Λογιστική Παλινδρόμηση

2.1 Ερώτημα Δ

Ο ταξινομητής μας έχει υλοποιηθεί στο αρχείο `logistic_regression.py`. Η πλήρης τεκμηρίωση του μοντέλου, των μεθόδων του και των υπερπαραμέτρων βρίσκεται εκεί με τη μορφή docstrings. Η κανονικοποίηση L_2 έχει ήδη υλοποιηθεί σε αυτό το αρχείο, αλλά για τους σκοπούς αυτής της ερώτησης θα θέσουμε την υπερπαράμετρο λ ως 0, παρακάμπτοντας την.

Ο κώδικας για την εκτέλεση του μοντέλου βρίσκεται στο αρχείο `run_logistic.py`. Θα τρέξουμε το μοντέλο με υπερπαραμέτρους `iter = 500` και `alpha = 0.2`. Το αποτέλεσμα είναι η ακρίβεια εκπαίδευσης να είναι ίση με 0.982 και η ακρίβεια ελέγχου ίση με 0.981. Τα πλήρη αποτελέσματα της εκπαίδευσης και του ελέγχου παρουσιάζονται στην Εικόνα 1.

2.2 Ερώτημα Ε

Επιλέγουμε το διάστημα των λ τιμών μας λογαριθμικά, εφόσον η βέλτιστη τιμή κανονικοποίησης είναι πολύ πιο πιθανό να βρίσκεται αρκετά κοντά στο 0. Η λογαριθμική κλίμακα μας επιτρέπει να ψάξουμε πιο πολλές τιμές του λ όσο πιο κοντά φτάνουμε στο κάτω όριο αναζήτησης μας, το 10^4. Η προσέγγιση αυτή χρησιμοποιείται και στην πράξη για κανονικοποίηση L^2 [1]

Σημειώνουμε ότι λόγω υπολογιστικών απαιτήσεων, ο αριθμός επαναλήψεων των μοντέλων μας μειώνεται στις 250 επαναλήψεις. Κατά την εκτέλεση του προγράμματος υπάρχει επίσης πιθανότητα να εμφανιστούν ειδοποιήσεις για αριθμητική υπερχείλιση. Αυτό είναι αποτέλεσμα της επιλογής πολύ μεγάλης τιμής του λ, κυρ-

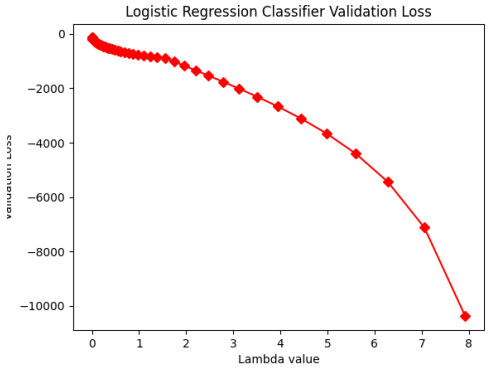

Figure 2: Τα αποτελέσματα της αναζήτησης για το βέλτιστο λ. Οι ρόμβοι αντιπροσωπεύουν τις τιμές που εξετάσαμε. Παρατηρείστε το πλήθος των τιμών που εξετάστηκαν στην αρχή συγκριτικά με το τέλος του πεδίου αναζήτησής μας.

ίως στο διάστημα [8, 10]. Σε αυτό το σημείο η κανονικοποίηση είναι τόσο ισχυρή που αποτρέπει το μοντέλο μας από το να μάθει, και έτσι αυτό μαντεύει την ίδια κατηγορία για κάθε παράδειγμα ελέγχου.

Στην δική μας περίπτωση το μοντέλο, κρατώντας τις υπόλοιπες υπερπαραμέτρους ίσες με το προηγούμενο υποερώτημα, προτιμά την ελάχιστη τιμή κανονικοποίησης $\lambda = 10^{-4}$ με ακρίβεια ελέγχου ίση με 0.981. Παρατηρούμε ότι η ακρίβεια ελέγχου με την επιλεγμένη τιμή λ είναι μικρότερη από την αντίστοιχη στο υποερώτημα Δ. Αυτό μας υποδεικνύει είτε ότι η βέλτιστη τιμή βρίσκεται έξω (και πιο συγκεκριμένα πριν) από το διάστημα αναζήτησης μας, είτε ότι για την διαφορά ευθύνεται το στατιστικό σφάλμα.

Ο κώδικας παραγωγής των παραπάνω γραφημάτων και αποτελεσμάτων βρίσκεται στο αρχείο `run_logistic.py`. Τα πλήρη αποτελέσματα αναζήτησης της υπερπαραμέτρου παρουσιάζονται στην Εικόνα 2.

3 ΜΕΡΟΣ Γ - Νευρωνικό Δίκτυο

3.1 Ερώτημα ΣΤ

Το νευρωνικό μας δίκτυο έχει υλοποιηθεί στο αρχείο `mlp.py`. Αποτελείται από δύο πίνακες βαρών και δύο πίνακες bias:

- Ο πίνακας h_w (hidden weights) έχει μέγεθος IxH
- Ο πίνακας o_w (output weights) έχει μέγεθος HxO
- Ο πίνακας h_b (hidden bias) έχει μέγεθος $1xH$
- Ο πίνακας o_b (output bias) έχει μέγεθος $1xO$

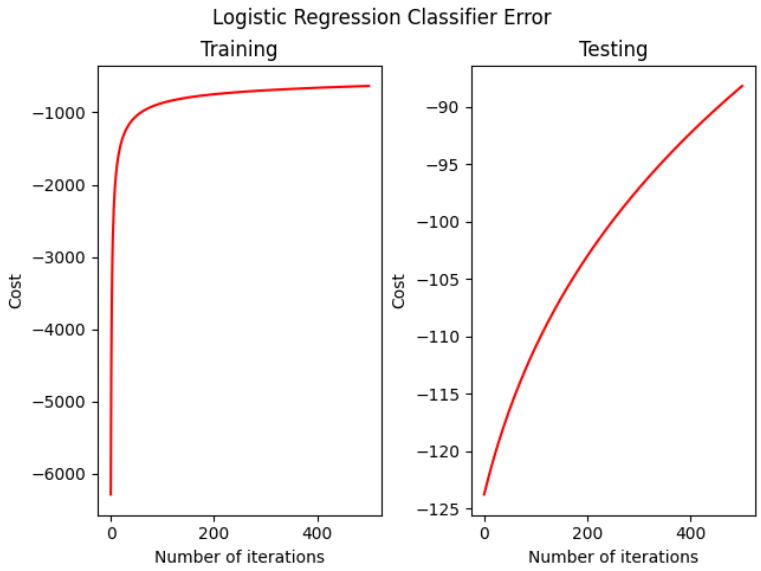

Figure 3: Το κόστος εκπαίδευσης ως συνάρτηση των επαναλήψεων του αλγορίθμου gradient descent.

όπου I=`input_size`, H=`hidden_layer_size`, O=`output_size`. Ο κώδικας για την εκτέλεση του δικτύου βρίσκεται στο αρχείο `run_mlp.py`, το οποίο εκτελεί κώδικα και για τα ερωτήματα Η, Θ.

Θα τρέξουμε το μοντέλο με υπερπαραμέτρους m=2, η=`0.2` και `tolerance`=`0.001`. Το `tolerance` είναι μια υπερ-παράμετρος απαραίτητη για το early stopping, και η οποία καθορίζει πόσο το κόστος πρέπει να έχει μειωθεί για να θεωρείται η τρέχουσα εποχή "βελτίωση". Το αποτέλεσμα είναι η ακρίβεια εκπαίδευσης να είναι ίση με 0.978, το τελικό κόστος εκπαίδευσης ίσο με 0.0671, η ακρίβεια ελέγχου ίση με 0.974 και το μέσο κόστος ελέγχου ίσο με 0.0734. Τα πλήρη αποτελέσματα της εκπαίδευσης παρουσιάζονται στην Εικόνα 3.

3.2 Ερώτημα Ζ

Ο τύπος της Δυαδικής Διασταυρούμενης Εντροπίας είναι:

$$E_b = -(t \ln \hat{y} + (1-t)\ln(1-\hat{y})) \tag{1}$$

όπου t το διάνυσμα των ετικετών δεδομένων και $\hat{y}$ το της εκτιμώμενης πιθανότητας.

Στην παραπάνω εξίσωση παρατηρούμε ότι η τιμή t είναι πάντοτε γνωστή, αντίθετα με τη $\hat{y}$, επομένως θα παραγωγίσουμε με βάση το $\hat{y}$. Εφόσον $y = f(x) + g(x) \iff y' = f'(x) + g'(x)$ ισχύει ότι:

$$\frac{\partial E_b}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}}(t \ln \hat{y}) + \frac{\partial}{\partial \hat{y}}(1-t)\ln(1-\hat{y}) \tag{2}$$

Αναλύουμε τις επιμέρους συναρτήσεις του αθροίσματος της παραγώγου:

$$\frac{\partial}{\partial \hat{y}}(t \ln \hat{y}) = \frac{\partial}{\hat{y}}(t \ln \hat{y}) + \ln \hat{y} \frac{\partial}{\partial \hat{y}} t = \frac{t}{\hat{y}} + \ln \hat{y} \cdot 0 = \frac{t}{\hat{y}} \tag{3}$$

$$\frac{\partial}{\partial \hat{y}}((t-1)\ln(1-\hat{y})) = (1-t)\frac{\partial}{\partial \hat{y}}(\ln(1-\hat{y})) + \ln(1-\hat{y})\frac{\partial}{\partial \hat{y}}(1-t) = \frac{1-t}{1-\hat{y}} + \ln(1-\hat{y}) \cdot 0 = \frac{1-t}{1-\hat{y}} \tag{4}$$

Επομένως, αθροίζοντας τις 3, 4, η 2 γίνεται:

$$\frac{\partial E_b}{\partial \hat{y}} = -(\frac{t}{\hat{y}} - \frac{1-t}{1-\hat{y}}) = (\frac{t}{\hat{y}} + \frac{1-t}{1-\hat{y}}) \tag{5}$$

Ο κώδικας επαλήθευσης του αναλυτικού τύπου βρίσκεται στο αρχείο `test_mlp.py`.

3.3 Ερώτημα Η

Εκτελούμε grid search για τις παραμέτρους m, η. Επιλέγουμε το διάστημα των η τιμών μας στο χώρο αναζήτησης $[0.5, 10^{-5}]$, εκθετικά κοντά στο 0.5 εφόσον εμπειρικά αναμένουμε ότι η βέλτιστη τιμή της θα βρίσκεται κοντά στις τιμές $0.5, 0.01, 0.01$. το οποίο για το συγκεκριμένο δίκτυο είναι το $(\eta = 0.5, m = 0.5, E = 172)$ με κόστος επικύρωσης ίσο με 0.0423.

Το πρόγραμμα εκτέλεσης του grid search βρίσκεται στο αρχείο `run_mlp.py`. Η αναζήτηση χρειάζεται περίπου 10 λεπτά για να βγάλει το βέλτιστο συνδυασμό υπερπαραμέτρων, το οποίο οφείλεται σχεδόν αποκλειστικά στην εκπαίδευση και επαλήθευση μοντέλων με $M \in 256, 512, 1024$. Λόγω των περιορισμών στις υπερπαραμέτρους δεν μπορούμε να μειώσουμε σημαντικά τον χρόνο εκτέλεσης.

3.4 Ερώτημα Θ

Χρησιμοποιούμε την μέθοδο `predict` του μοντέλου μας για να αποκτήσουμε τις προβλεπόμενες ετικέτες του. Η μέθοδος αυτή χρησιμοποιεί μεθόδους της numpy, ισοδύναμες της κατηγοριοποιήσης με βρόγχο. Στο σύνολο ελέγχου το μοντέλο με τις βελτιστοποιημένες παραμέτρους επιτυγχάνει ακρίβεια ελέγχου ίση με 0.979 και κόστος ελέγχου ίσο με 0.0569.

3.5 Ερώτημα Ι

Ο κώδικας του ταξινομητή SGD μπορεί να βρεθεί στο αρχείο `sgd.py`.

Υλοποιούμε τον ταξινομητή στοχαστικής καθοδικής κλίσης ως υποκλάση του προηγούμενού μας ταξινομητή. Η κύρια αλλαγή είναι το πέρασμα mini-batches στην μέθοδο `backpropagation` αντί για το σύνολο των δειγμάτων εκπαίδευσης. Πιο συγκεκριμένα, αντί να περάσουμε στην `backpropagation` έναν πίνακα διαστάσεων 9072x784, περνάμε πίνακα Bx784 όπου B το μέγεθος του mini-batch. Τα παραδείγματα εκπαίδευσης και οι ετικέτες τους ανακατεύονται πριν κάθε επανάληψη του αλγορίθμου έτσι ώστε το μοντέλο μας να εξετάζει κάθε φορά διαφορετικό σύνολο παραδειγμάτων. Η ίδια η μέθοδος `backpropagation` παραμένει ίδια.

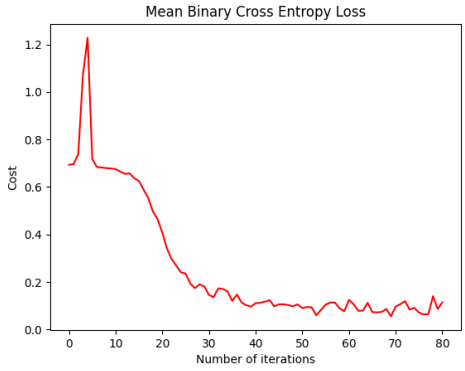

Figure 4: Το κόστος εκπαίδευσης ως συνάρτηση των επαναλήψεων του αλγορίθμου stochastic gradient descent.

Το πρόγραμμα εκτέλεσης του ταξινομητή καθώς και των παρακάτω grid searches βρίσκεται στο αρχείο `run_sgd.py`. Τρέχοντας τον ταξινομητή μας με παρόμοιες παραμέτρους με τον προηγούμενο και με αυθαίρετο μέγεθος mini-batch=128, επιτυγχάνει ακρίβεια εκπαίδευσης 0.967 με κόστος 0.0817 και ακρίβεια ελέγχου 0.965 με κόστος 0.1194. Η ακρίβεια αυτή είναι κατά μικρό βαθμό χειρότερη από τον ταξινομητή λογιστικής παλινδρόμησης, αν και λόγω του υπολογισμού με mini-batches και του κριτηρίου early-stopping, το μοντέλο είναι σημαντικά γρηγορότερο.

Ένα πλήρες γράφημα για την εξέλιξη του κόστους εκπαίδευσης βρίσκεται στην Εικόνα 4. Παρατηρούμε ότι το κόστος του ταξινομητή μας ακολουθεί την ίδια πορεία με αυτό της Εικόνας 3, αν και είναι εμφανώς περισσότερο ανώμαλη. Αυτό οφείλεται στην τυχαιότητα του αλγόριθμου SGD, ο οποίος προσθέτει "θόρυβο" στη διαδικασία εκπαίδευσης μέσω του τυχαίου και περιορισμένου πλήθους δεδομένων που βλέπει το μοντέλο μας σε κάθε επανάληψη του SGD.

Δοκιμάζοντας το μοντέλο μας με τιμές $B = 2^i, i = 1, 2, .., 8$, η βέλτιστη τιμή μεγέθους mini-batch φαίνεται να είναι η $B = 32$, με κόστος επικύρωσης ίσο με 0.1215 και με $E = 52$ εποχές εκπαίδευσης. Χρησιμοποιώντας τη βέλτιστη τιμή Β, μπορούμε να τρέξουμε την ίδια μέθοδο αναζήτησης βέλτιστων υπερπαραμέτρων όπως στο ερώτημα Η. Με την ίδια μεθοδολογία και το ίδιο χώρο αναζήτησης υπερπαραμέτρων, το πρόγραμμα μας βρίσκει βέλτιστες υπερ-παραμέτρους ($\eta = 0.139, m = 32, E = 57$). Δυστυχώς το βέλτιστο αυτό μοντέλο φαίνεται να πάσχει είτε από overfitting είτε από κάποιο αριθμητικό σφάλμα, καθώς η ακρίβεια ελέγχου του είναι 0.618, με κόστος 0.6304.

Σημειώνουμε ότι λόγω της τυχαιότητας του αλγορίθμου SGD τα αποτελέσματα της εκτέλεσης μπορεί να είναι διαφορετικά από αυτά που αναγράφονται στο έγγραφο αυτό.

References

[1] Jerome Friedman, Trevor Hastie and Robert Tibshirani. 'Regularization Paths for Generalized Linear Models via Coordinate Descent'. In: *Journal of Statistical Software* (2010).