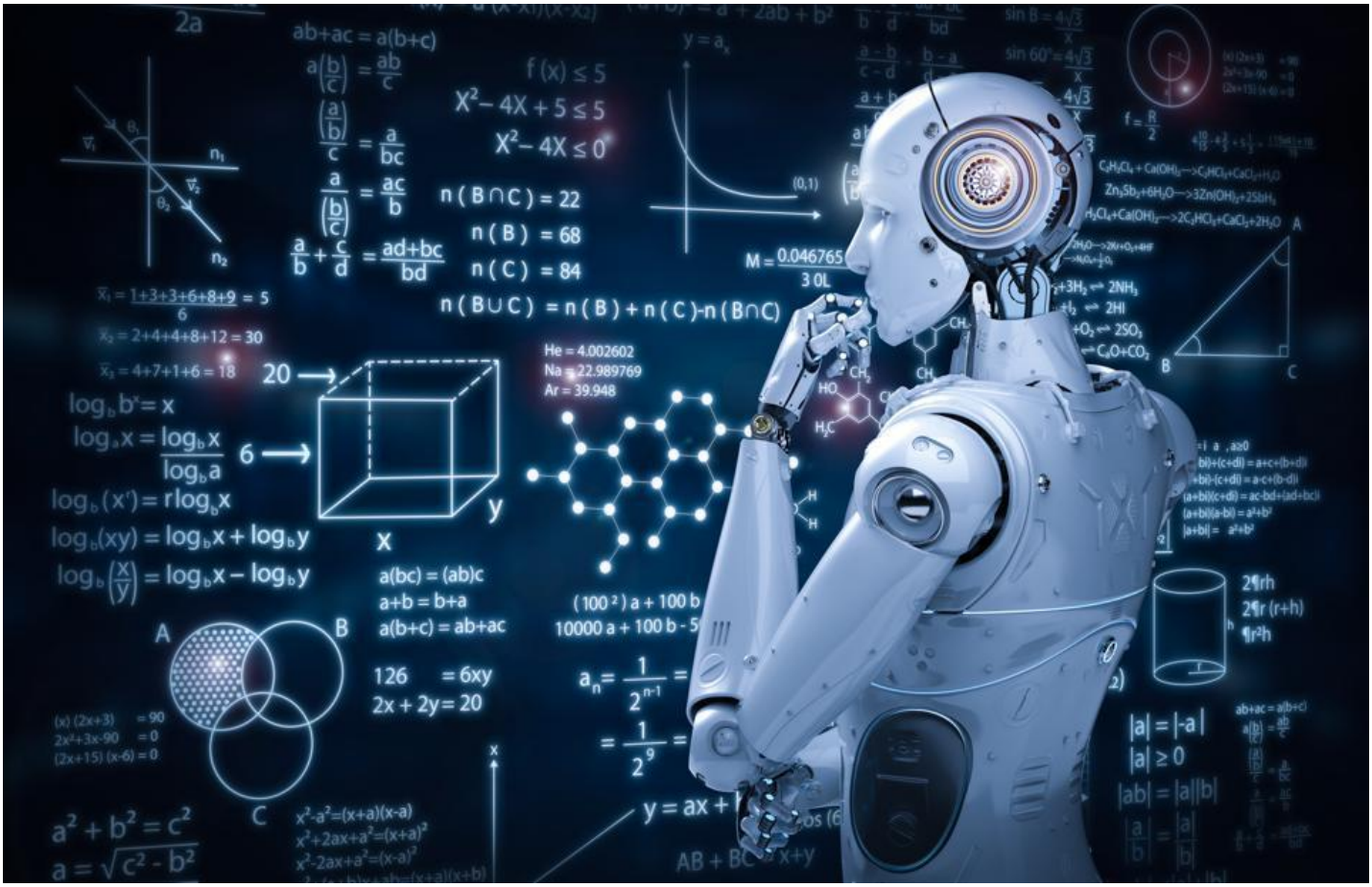
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Machine Learning

Project: MNIST digit classification



[1]

Θανάς Κούρο,
3170078

[1] https://imageio.forbes.com/specials-images/dam/imageserve/966248982/960x0.jpg?format=jpg&width=960

ΜΕΡΟΣ Α: Pre-proccessing

a) Υλοποιήθηκε.
b) Υλοποιήθηκε.

Η υλοποίηση βρίσκεται στο αρχείο PartA.ipynb, με σχόλια για κάθε κελί του notebook.
Στο αρχείο LoadDatasets.ipynb, υπάρχει μία ενιαία μέθοδος που υλοποιεί τα 2 ζητούμενα και επιστρέφει τα κατάλληλα datasets. Το LoadDatasets.ipynb θα γίνει import από τα μέρη Β, Γ και θα κληθεί η μέθοδος για την αρχικοποίηση των datasets.

ΜΕΡΟΣ Β: Logistic Regression

Η υλοποίηση του αλγορίθμου υπάρχει στο αρχείο LogisticRegression.ipynb, όπου έχει δημιουργηθεί η κλάση LogisticRegression. Περιέχει την μέθοδο fit για την εκπαίδευση, και μέθοδο για ενημέρωση βαρών με gradient ascent, υπολογισμό accuracy και τη sigmoid function.

Η υλοποίηση των ερωτημάτων βρίσκεται στο αρχείο PartB.ipynb, όπου γίνεται import το LogisticRegression.ipynb για την δημιουργία των μοντέλων μέσω της κλάσης LogisticRegression.

c) class Logistic Regression:
- Στη μέθοδο fit, γίνεται αρχικοποίηση των βαρών W και του bias term b. Χρησιμοποιείται vectorization για την εκπαίδευση και δεν γίνεται iteration σε κάθε training example. Καλείται ο αλγόριθμος gradient ascent και γίνεται η εκπαίδευση για τον αριθμό των εποχών που έχει δοθεί ως όρισμα. Στο τέλος κάθε εποχής ενημερώνεται ο πίνακας βαρών και το bias term.
- Στη μέθοδο __computeCostGrads, υλοποιείται ο αλγόριθμος gradient ascent, είτε με L2 regularization είτε χωρίς, ανάλογα το όρισμα της υπερπαραμέτρου lambda.

d) Στο PartB.ipynb δημιουργούμε ένα instance της κλάσης Logistic Regression και εκπαιδεύουμε το μοντέλο για 100 εποχές και learning rate = 1**.** Το **test accuracy** που προκύπτει στα test data ισούνται με **97,95%.**

e) Έχει προστεθεί υπερπαράμετρος κανονικοποίησης για L2 regularization στη μέθοδο __computeCostGrads και αν έχει τιμή διάφορη του μηδενός εφαρμόζεται κανονικοποίηση κατά τον υπολογισμό των gradients.
Η αρχικοποίηση των 100 διαφορετικών τιμών lambda γίνεται στο PartB.ipynb. Για να εξαπλώσουμε καλύτερα τις τιμές σε όλο το διάστημα, αρχικοποιούμε τις τιμές τυχαία, ομοιόμορφα στα παρακάτω διαστήματα:
- $[10^{-4} - 10^{-3}]$
- $[10^{-3} - 10^{-2}]$
- $[10^{-2} - 10^{-1}]$
- $[10^{-1} - 10^{-0}]$

Η καλύτερη τιμή του **lambda** που μεγιστοποιεί το accuracy στα development data είναι **0.006351** και το **test accuracy** που προκύπτει είναι ίο με **97,95%**, όπως και στο ερώτημα d.

ΜΕΡΟΣ Γ: Neural Networks

Η υλοποίηση των ερωτημάτων βρίσκεται στο αρχείο PartC.ipynb, όπου γίνεται import το NeuralNetworks.ipynb για την δημιουργία των μοντέλων.

f) Η υλοποίηση του αλγορίθμου υπάρχει στο αρχείο NeuralNetworks.ipynb, όπου έχουν δημιουργηθεί οι κλάσεις Layer, NeuralNetwork.
Αρχικά, υλοποιούμε τη sigmoid function και την παράγωγό της, καθώς και την leaky ReLu(προαιρετικά) με την παράγωγό της.

- class Layer: Κάθε instance αποτελεί ένα hidden layer ή output layer.
  Στην αρχικοποίηση ενός layer, δίνονται σαν παράμετροι ο αριθμός των νευρώνων του Layer και ο αριθμός βαρών για κάθε νευρώνα. Τέλος, δίνεται η συνάρτηση ενεργοποίησης (sigmoid/lealy relu).
  Τα βάρη αρχικοποιούνται τυχαία, πολλαπλασιαζόμενα με τον παράγοντα $\sqrt{\frac{1}{weights-1}}$, για την αποφυγή exploding gradients.
  Η μέθοδος forward_prop εκτελεί το forward propagation και η back_prop το back propagation.
- class NeuralNetwork: Αποτελεί το μοντέλο με τις κατάλληλες υπερπαραμέτρους.
  Η μέθοδος input_layer καλείται μία φορά και προστίθεται το training dataset με τις αντίστοιχες σωστές κατηγορίες y.
  Η add_layer προσθέτει από ένα hidden layer σε κάθε κήση της ή το output_layer.
  Η fit καλείται για την εκπαίδευση του μοντέλου με τα given parameters. Καλείται η __forward_prop() και η __back_prop() σε κάθε επανάληψη.
- Η __back_prop(), αρχικά υπολογίζει τα gradients για το output layer, καθώς διαφοροποιείται από τον υπολογισμό των gradients των hidden layers.
  Έπειτα καλείται η back_prop() Της κλάσης Layer για όλα τα hidden layers.

g) $L = y_i \cdot \log(a) + (1 - y_i) \cdot \log(1 - a)$ **, όπου i το i-οστό training example**
$\frac{dL}{da} = \frac{y_i}{a} - \frac{1-y_i}{1-a}$**, όπου** $a = \frac{1}{1+e^{-z}}$
$\frac{dL}{dz} = \frac{dL}{da} \cdot \frac{da}{dz} = = \frac{dL}{da} \cdot \frac{-e^{-z}}{1\pm e^{-z}} \cdot \frac{1}{1\pm e^{-z}} = \left(\frac{y_i}{a} - \frac{1-y_i}{1-a}\right) \cdot (1-a) \cdot a = y_i - a$
Επίσης, προστέθηκε στην fit της NeuralNetwork class, early stopping parameter. Εάν τεθεί true, τότε ελέγχει αν το dev loss αυξάνεται ή παραμένει σταθερό για 5 εποχές και τερματίζει. Θέτουμε ως ελάχιστο αριθμό εποχών το 10 και από εκεί και πέρα ελέγχεται η συνθήκη. Ως μέγιστο αριθμό εποχών θέτουμε το 500.
Εκπαιδεύοντας το μοντέλο με early stopping στο αρχείο PartC.ipynb, το βέλτιστο μοντέλο προκύπτει σε **214 εποχές**, με **learning rate = 0.1**, **M=8** (αριθμός νευρώνων κρυμμένου επιπέδου) και test accuracy **97.46%.**

h) Εξαπλώνουμε τις τυχαίες τιμές του learning rate ομοιόμορφα στο ζητούμενο διάστημα.
Αρχικοποιούμε και τις διαφορετικές πιθανές τιμές του M.
Οι υπερ-παράμετροι του βέλτιστου μοντέλου είναι:
learning rate: 0.70960
M: 4
epochs: 31

i) **test accuracy = 96.34%.**

j) Αρχικοποιούμε τον πίνακα B με τις διαφορετικές τιμές των batch size.
1. Τρέχοντας το αρχικό μοντέλο του ερωτήματος g (ή ζ στην εκφώνηση) προκύπτει βέλτιστο **batch size = 128** και **epochs =5**. Το **test accuracy** που που πρκύπτει είναι **97,08%**.
2. Έχει υλοποιηθεί ο κώδικας για την εύρεση των υπολοίπων υπερπαραμέτρων του ερωτήματος h (θ στην εκφώνηση)**.**