ΕΘΝΙΚΟ ΚΑΙ
ΚΑΠΟΔΙΣΤΡΙΑΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ

Τεχνητή Νοημοσύνη **2**

1η Εργασία

Ιωάννης Δαλιάνης 1115201700027

Τμήμα Πληροφορικής και Τηλεπικοινωνιών, ΕΚΠΑ

Οκτώβριος 2020

Περιεχόμενα

Σελίδα

1 Θεωρητική Άσκηση **2**

1.1 Πρώτη εξίσωση 2

1.2 Δεύτερη εξίσωση 3

2 Θεωρητική Άσκηση **4**

3 Θεωρητική Άσκηση **5**

4 Θεωρητική Άσκηση **6**

5 Προγραμματιστική Άσκηση **9**

5.1 FNN_GLOVE.ipynb 9

5.2 TRY_FFNN.ipynb 9

5.3 AI2_FFNN.ipynb 10

5.4 AI2_NOT_FFNN.ipynb 11

5.5 Συμπέρασμα 11

1 Θεωρητική Άσκηση

1.1 Πρώτη εξίσωση

Έχουμε τη γραμμική συνάρτηση:

$$y(x) = w^T \cdot x + w_0$$

από την ενότητα 4.1.1 του βιβλίου Patter Recognition and Machine Learning το οποίο βρίσκεται εδώ. Το w είναι το weight vector και $\|w\| \in \mathbb{R}$ με $\|w\| \neq 0$.

Το διάνυσμα βαρών w είναι κάθετο στη γραμμή απόφασης(decision boundary). Έστω $\vec{x}$ το διάνυσμα από την αρχή του επιπέδου στο x. Η προβολή του διανύσματος $\vec{x}$ στο w θα έχει μέγεθος ίσο με την ορθογώνια απόσταση από τη γραμμή απόφασης. Συμβολίζεται με:

$$\|proj_w \vec{x}\| = \frac{w \cdot \vec{x}}{\|w\|}$$

Πηγή: `https://en.wikibooks.org/wiki/Linear_Algebra/Orthogonal_Projection_Onto_a_Line` Εφόσον το x είναι στη γραμμή, σημαίνει ότι

$$0 = w^T \cdot x + w_0$$

Οπότε εν τέλει έχουμε ότι:

$$r = \frac{w^T \cdot x}{\|w\|} = \frac{-w_0}{\|w\|}$$

1.2 Δεύτερη εξίσωση

Έχουμε την εξίσωση:

$$x = x_\perp + r \cdot \frac{w}{\|w\|}$$

Κάνουμε ανάλυση του x σε στοιχεία σε σχέση με τη γραμμή απόφασης: το $x_\perp$ στη γραμμή και κάτι κάθετο στη γραμμή. Όντας κάθετα στο όριο, είναι προς την κατεύθυνση του w, αλλά δεν ξέρουμε πόσο μακριά, οπότε εισάγουμε το μήκος ως το άγνωστο r. Τώρα, όπως αναφέραμε, το y είναι μηδέν στο όριο, έτσι έχουν $y(x_\perp) = 0$. Έχουμε:

$$x = x_\perp + r \cdot \frac{w}{\|w\|}$$

$$w^T x + w_0 = w^T x_\perp + w_0 + r\frac{w^T w}{\|w\|}$$

$$y(x) = y(x_\perp) + r\frac{\|w\|^2}{\|w\|}$$

$$y(x) = 0 + r \cdot \|w\|$$

και έτσι:

$$r = \frac{y(x)}{\|w\|}$$

2 Θεωρητική Άσκηση

Έχουμε $z = Wx$ όπου $W \in \mathbb{R}^{n \times m}$. Η Jacobian του z θα είναι διαστάσεων $n \times m$.

$$z_i = \sum_{k=1}^{m} W_{ik} x_k$$

Οπότε ένα στοιχείο $(\frac{\partial z}{\partial x})_{ij}$ της Jacobian θα είναι:

$$(\frac{\partial z}{\partial x})_{ij} = \frac{\partial z_i}{\partial x_j} = \frac{\partial}{\partial x_j} \sum_{k=1}^{m} W_{ik} x_k = \sum_{k=1}^{m} W_{ik} \frac{\partial}{\partial x_j} x_k = W_{ij}$$

επειδή $\frac{\partial}{\partial x_j} x_k = 1$ αν $k = j$ και 0 αλλιώς. Έτσι έχουμε $\frac{\partial z}{\partial x} = W$.

3 Θεωρητική Άσκηση

$\widehat{y} = \sigma(x^T w)$ Το x και το w είναι column vectors. Το x^T είναι row vector.

$$\frac{\partial \widehat{y}}{\partial w} = \frac{\partial(\frac{1}{1+e^{-w^T x}})}{\partial w}$$

έστω $u = w^T x$ οπότε $\partial(\frac{1}{1+e^{-w^T x}}) = \partial(\frac{1}{1+e^{-u}})$
έχουμε:

$$\partial(\frac{1}{1+e^{-u}}) = (\frac{1}{1+e^{-u}})' = -\frac{(1+e^{-u})'}{(1+e^{-u})^2} = -\frac{(e^{-u})'}{(1+e^{-u})^2} = -\frac{(-u)'(e^{-u})}{(1+e^{-u})^2}$$

όπου $(u)' = (w^T x)'$

4 Θεωρητική Άσκηση

Από το computational graph έχουμε:

$$\begin{aligned} x &= 2 \\ y &= 3 \\ z &= 4 \\ h &= x + y = 5 \\ i &= x * y = 6 \\ g &= h + i = 11 \\ f &= g * z = 44 \end{aligned}$$

Forward propagation:

$$\begin{aligned} f(x,y,z) &= ((x+y)+(x*y))*z \\ f(2,3,4) &= ((2+3)+(2*3))*4 = 44 \end{aligned}$$

Backward propagation:

$$\frac{\partial h}{\partial x} = \frac{\partial (x+y)}{\partial x} = y = 3$$

$$\frac{\partial h}{\partial y} = \frac{\partial (x+y)}{\partial y} = x = 2$$

$$\frac{\partial i}{\partial x} = \frac{\partial (x*y)}{\partial x} = y = 3$$

$$\frac{\partial i}{\partial y} = \frac{\partial (x*y)}{\partial y} = x = 2$$

$$\frac{\partial f}{\partial f} = 1$$

$$\frac{\partial f}{\partial z} = \frac{\partial f}{\partial f}\frac{\partial (g*z)}{\partial z} = g = 11$$

$$\frac{\partial f}{\partial g} = \frac{\partial f}{\partial f}\frac{\partial (g*z)}{\partial g} = z = 4$$

$$\frac{\partial f}{\partial h} = \frac{\partial (g*z)}{\partial h} = \frac{\partial ((h+i)*z)}{\partial h} = \frac{\partial (h*z+i*z)}{\partial h} = z = 4$$

$$\frac{\partial f}{\partial i} = \frac{\partial (g * z)}{\partial i} = \frac{\partial ((h+i) * z)}{\partial i} = \frac{\partial (h * z + i * z)}{\partial i} = z = 4$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial h}\frac{\partial h}{\partial x} + \frac{\partial f}{\partial i}\frac{\partial i}{\partial x} = 4 \cdot 1 + 4 \cdot 3 = 16$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial h}\frac{\partial h}{\partial y} + \frac{\partial f}{\partial i}\frac{\partial i}{\partial y} = 4 \cdot 1 + 4 \cdot 2 = 12$$

5 Προγραμματιστική Άσκηση

Παραδίδονται 4 αρχεία python notebook:

5.1 FNN_GLOVE.ipynb

Χρησιμοποιούμε τα δεδομένα του Glove αρχείου για να φτιάξουμε ένα corpus το οποίο θα χρησιμοποιηθεί μετέπειτα. Κάνουμε tokenize τα tweets και split τα δεδομένα. Για το training χρησιμοποιείται ένα σχετικά μικρό κομμάτι δεδομένων γιατί το κομμάτι της εκπαίδευσης παίρνει αρκετή ώρα να εκτελεστεί. Για το FFNN χρησιμοποιούνται 3 linear layers με ReLU activation και για την έξοδο των αποτελεσμάτων χρησιμοποιείται η SoftMax. Η συνάρτηση make_bow_vector ψάχνει για κάθε λέξη μιας πρότασης αν εμφανίζεται στο dictionary και προσαυξάνει τη συχνότητά της γυρνώντας ένα torch vector για κάθε πρόταση ανάλογα με την "αξία" των λέξεων, δηλαδή με βάση τη συχνότητά τους. Το μοντέλο εκπαιδεύεται με διάσταση εισόδου ίδια με το μέγεθος του λεξιλογίου που έχουμε δημιουργήσει(review_dict), με hidden dimension 500, output 2 διότι τα labels είναι 0 και 1. Οι εποχές εκπαίδευσης είανι 10 και το λεξιλόγιο που δημιουργήθηκε από το glove χρησιμοποιείται ως φίλτρο για την αριθμητική αξία κάθε πρότασης. Τα losses σώζονται σε ένα csv αρχείο το οποίο ανοίγεται αργότερα για ploting. Το αποτέλεσμά μας είναι ένα accuracy score γύρω στο 0.68.

5.2 TRY_FFNN.ipynb

Στο μεγαλύτερό του κομμάτι ακολουθεί την ίδια υλοποίηση με το αρχείο FNN_GLOVE.ipynb. Κατασκευάζεται ένα FFNN στο οποίο χρησιμοποιείται για την NLP επεξεργασία το πακέτο corpora από τη βιβλιοθήκη gensim, το οποίο προσφέρει ένα tokenizer

εργαλείο για την ειδική επεξεργασία που κάνει στα κείμενα στην οποία πραγματοποιεί και padding για να φτιάξει το dictionary. Πρόκειται γενικά για ένα αργό μοντέλο το οποίο για hidden dimension 500, output 2 και 15 epochs χρειάζεται 85 λεπτά και εν τέλει καταφέρνουμε ένα accuracy score της τάξης του 0.62. Επαναλαμβάνουμε το μοντέλο για διαφορετικές hyperparameters. Χρησιμοποιείται sigmoid ωςσυνάρτηση ενεργοποίησης, ενώ εκπαιδεύεται για 15 epochs όπως και το αμέσως προηγούμενο μοντέλο. Χρειάζεται 366 λεπτά και εν τέλει καταφέρνουμε ένα accuracy score της τάξης του 0.71.

5.3 AI2_FFNN.ipynb

Κατασκευάζεται ένα FFNN στο οποίο χρησιμοποιείται για την NLP επεξεργασία το προεκπαιδευμένο μοντέλο BERT της Google από τη βιβλιοθήκη Hugging Face. Το BERT μοντέλο είναι εκπαιδευμένο πάνω σε ολόκληρες προτάσεις. Η βιβλιοθήκη BERT μας προσφέρει ένα tokenizer εργαλείο για την ειδική επεξεργασία που κάνει στα κείμενα στην οποία χρησιμοποιεί και padding με τα ειδικά tokens [CLS] [SEP]. Δημιουργούμε ένα iterator για τα δεδομένα μας με τη DataLoader κλάση της PyTorch. Το BERT χρησιμοποιεί 12 transformer layers καθένα από τα οποία το καθένα παίρνει μια λίστα από token embeddings και παράγει τον ίδιο αριθμό από embeddings. Παρέχει την κλάση BertForSequenceClassification για classification προβλήματα, την οποία επεκτείνουμε θέτοντας hidden layers 768 και συνάρτηση ενεργοποίησης ReLU. Για την κλάση BertForSequenceClassification οι δημιουργοί προτείνουν Batch size 16 ή 32, adam opimizer με learning rate 5e-5 ή 3e-5 ή 2e-5 και 2, 3 ή 4 εποχές. Εκτυπώνεται το roc auc plot και βλέπουμε ότι έχουμε ένα accuracy score 0.75 σε 21 λεπτά. Ξαναεκπαιδεύουμε το μοντέλο δίνοντάς του SIGMOID ως συνάρτηση ενεργοποίησης και hidden size του classifier 60. Έχουμε ένα accuracy score 0.75 πάλι σε 20 λεπτά.

5.4 AI2_NOT_FFNN.ipynb

Recursive και Convolutional Neural Network χρησιμοποιώντας glove τα οποία τα έκανα στην αρχή επειδή είχα ξεχάσει ότι η άσκηση ζητούσε αποκλειστικά FFNN. Είπα εφόσον τα έκανα να τα στείλω αλλά δε χρειάζεται να ληφθούν υπόψη.

5.5 Συμπέρασμα

Η καλύτερη επίδοση έφτασε την επίδοση της προηγούμενης εργασίας. Με μεγαλύτερη υπολογιστική ισχύ ή μεγαλύτερη υπομονή για εκπαίδευση των μοντέλων πάνω σε ολόκληρο το dataset θα μπορούσαμε να έχουμε καλύτερα αποτελέσματα. Σε γενικές γραμμές ο ADAM optimizer και η ReLU activation function είναι οι καλύτερες επιλογές για το binary classification.