**ΕΠΛ 442: Μηχανική Μάθηση**

**ΕΡΓΑΣΙΑ 3: Letter Classification  
25/10/2023**

**Νικόλαος Θεοδώρου 1030496  
  
Εισαγωγή**

Σε αυτήν την αναφορά περιγράφονται οι λεπτομέρειες της εκπαίδευσης ενός νευρωνικού δικτύου για την επίλυση του προβλήματος ταξηνομισης γραμμάτων του αγγλικού αλφαβήτου. Περιλαμβάνονται οι χρησιμοποιηθείσες μεταβλητές, οι επιλογές που αφορούν τη δομή του νευρωνικού δικτύου και τα αποτελέσματα που προέκυψαν από την εκπαίδευση. Η εργασία τρέχει με την εντολή python3 Trainer.py και απαιτεί python 3.6 η νεότερη και τα πακέτα numpy και matplotlib και έχει δοκιμαστεί σε περιβάλλον Linux/Debian.

**Προετοιμασία δεδομένων εισόδου**

Αρχικα τα δεδομένα θα πρέπει να κανονιικοποιηθούν πριν μπουν στο νευρωνικο δύκτιο. Χρησιμοποίησα την μέθοδο κανονικοποίσης MinMax για να κανονικοποιήσω κάθε στύλη των δεδομένων.  
Στη συνέχεια οργάνωσα τα δεδομένα ανά γράμμα που υπάρχει στην έξοδο.   
Μετά, μοίρασα τα ¾ των δεδομένων στο train set και το ¼ στο test set.  
Τέλος, ανακάτεψα τις εγγραφές των κάθε set δεδομένων.

Η πιο πάνω διαδικασία γίνεται με τη βοήθεια του κώδικα που έγραψα στα αρχεία FileManager.py και utility.py

Αυτή η διαδικασία γίνεται αυτόματα με το τρέξιμο του script Trainer.py, πριν να εκκινήσει η εκπαίδευση.

**Μεταβλητές και Επιλογές για το Νευρωνικό Δίκτυο**

Για την επίλυση του προβλήματος της εργασίας υλοποίησα ένα νευρωνικό δύκτιο SOM, ακριβώς όπως περιγράφεται από τον Kohonen. Δηλαδή, έβαλα 16 νευρώνες εισόδου όσο και το μέγεθος της εισόδου των δεδομένων και ένωσα τον κάθε νευρώνα εισόδου με όλους τους κρυφούς νευρώνες του χάρτη.

Αρχικά έγινε αρχικοποίηση των βαρών με τυχαίο τρόπο και με σταθερό seed, αφού βοηθά να σπάσει η συμμετρία και να αποφευχθεί το πρόβλημα των πανομοιότυπων κρυφών νευρώνων. Εάν τα βάρη αρχικοποιηθούν στην ίδια τιμή, όπως το μηδέν, τότε όλες οι κρυφές μονάδες θα λάβουν το ίδιο σήμα και θα εκτελέσουν τον ίδιο υπολογισμό, ανεξάρτητα από την είσοδο. Αυτό θα κάνει το δίκτυο να μην μπορεί να μάθει κάποια σημαντικά χαρακτηριστικά από τα δεδομένα. Η τυχαία αρχικοποίηση των βαρών διασφαλίζει ότι ο κάθε κρυφός νευρώνας εκτελεί διαφορετικό υπολογισμό και ανταποκρίνεται διαφορετικά στην είσοδο, γεγονός που επιτρέπει στο δίκτυο να μάθει διάφορα και χρήσιμα χαρακτηριστικά.

Για τον έλεγχο της απόδοσης του δυκτίου, πάντα έβλεπα ότι το clustering.txt έβγαζε λογικές ταξινομήσεις και στη συνέχεια υπολόγισα το quantum error στο test set.

Για την τοπολογία του δυκτίου είναι αναμφισβήτητο πως θα πρέπει να βάλουμε 16 νευρώνες εισόδου, ίσο με τον αριθμό των εισόδων που έχουμε στα δεδομένα μας και πάνω από 26 νευρώνες στο χάρτη, ίσο με τον αριθμό των γραμμάτων του αγγλικού αλφαβήτου.

Για την τοπολογία των κρυφών επιπέδων έγιναν δοκιμές με διαστάσεις χάρτη 10x10, 25x25, 50x50, 100x100 και 150x150. Παρατήρησα πως το πρόβλημα δεν λύνεται ικανοποιητικά με χάρτη 10x10 και πως το σφάλμα επιπεδώνεται με χάρτη 50x50 ή μεγαλύτερο κοντά στο 0.132. Ότιδήποτε πάνω από 100x100, φάνηκε να είνα υπερβολικό. Οι παρατηρήσεις έγιναν σύμφωνα με τα πιο κάτω αποτελέσματα:

|  |  |
| --- | --- |
| LR\_0.8 GS\_100 EP\_500 | 0.131924 |
| LR\_0.8 GS\_150 EP\_500 | 0.13304 |
| LR\_0.8 GS\_50 EP\_500 | 0.137194 |
| LR\_0.8 GS\_25 EP\_500 | 0.137582 |
| LR\_0.8 GS\_10 EP\_500 | 0.147515 |

Αποτελέσματα διάφορων τοπολογιών για ρυθμό μάθησης 0,8 και 500 εποχές

Έγινε δοκιμή διάφορων παραμέτρων του νευρωνικου δυκτίου και για το learning. Για το learning rate επέλεξα 0.2, αφού δίνει ιδανική αναλογία χρόνου εκπαίδευσης και ακρίβειας. Ακόμη σε σύκγριση και με τον πιο πάνω πίνακα για τοπολογία 50x50, το 0.2 learning rate, είχε καλύτερα αποτελέσματα από το 0.8 σε μόλις 30 εποχές αντι για 500.

|  |  |
| --- | --- |
| LR\_0.2 GS\_50 EP\_30 | 0.129627 |
| LR\_0.4 GS\_50 EP\_30 | 0.132978 |
| LR\_0.8 GS\_50 EP\_30 | 0.138277 |

Αποτελέσματα διάφορων ρυθμών μάθησης για τοπολογία 50x50 και 30 εποχές

Επίσης πρόσεξα πως με αυτό το ρυθμό μετά από 70 εποχές το error rate επιπεδόνεται.

|  |  |
| --- | --- |
| LR\_0.2 GS\_50 EP\_85 | 0.123475 |
| LR\_0.2 GS\_50 EP\_70 | 0.125845 |
| LR\_0.2 GS\_50 EP\_50 | 0.127899 |
| LR\_0.2 GS\_50 EP\_30 | 0.129627 |
| LR\_0.2 GS\_50 EP\_15 | 0.132327 |
| LR\_0.2 GS\_50 EP\_5 | 0.145242 |

Αποτελέσματα διάφορων εποχών με 0,2 ρυθμό μάθησης για τοπολογία 50x50

Αποτελέσματα Quantum Error στο test set, όλων των δοκιμών που έκανα

Νικητής με πιο χαμηλό quantum error rate στο test set, ήταν η διάταξη 50x50 με ρυθμό μάθησης 0,2 για 85 εποχές με 0,1234. Με λίγες εποχές και συγκριτικά μικρό μέγεθος χάρτη, μπορούμε να πούμε πως είναι και η πιο αποδοτική επιλογή ως προς την ακρίβεια ανά πόρους και επιτυγχάνει και την καλύτερη ακρίβεια.

Τέλος σύφωνα με τα πιο πάνω αποτελέσματα, οι επιλογές που έγιναν για τη δομή του νευρωνικού δικτύου είναι οι εξής:

* Πρώτο Επίπεδο: 16 νευρώνες (16 είσοδοι)
* Κρυφό Επίπεδο: Πίνακας μεγέθους 50x50
* Ρυθμός Μάθησης (learningRate): 0.2
* Αριθμός Επαναλήψεων (maxIterations): 85

**Ενδεικτικά δεδομένα αρχείου all\_data.txt:**

Out in1 in2 in3 in4 in5 in6 in7 in8 in9 in10 in11 in12 in13 in14 in15 in16

T,2,8,3,5,1,8,13,0,6,6,10,8,0,8,0,8

I,5,12,3,7,2,10,5,5,4,13,3,9,2,8,4,10

**Έξοδος**

Η έξοδος γινεται στα αρχεία “*results/clustering\_LR={learning rate}\_GS={grid size}\_EP={Epochs}.txt*” με το κάθε ένα να έχει τις ανάλογες στήλες για epoch training/test quantum error rate και την απεικόνηση του χάρτη.

**Ανάλυση Αποτελεσμάτων**

Σύμφωνα με τις πιο κάτω γραφική του results.txt και απεικονήση του clustering.txt, οι προβλεπόμενες εξόδοι σχετίζονται σωστά με τις εισόδους, καταφέρνοντας να επιτύχει το επιθυμητό αποτέλεσμα για όλα τα παραδείγματα του συνόλου δεδομένων.

Τα αποτελέσματα δείχνουν ότι το νευρωνικό δίκτυο κατάφερε να μάθει να ταξινομεί τα γράμματα. Οι μεταβλητές του δικτύου ενημερώθηκαν με βάση την αλγοριθμική διαδικασία του Kohonen. Δεν φαίνεται να κολλούμε σε τοπικά ελάχιστα.

**Γραφική Παράσταση**

Παρακάτω παρουσιάζονται γραφικές παραστάσεις για τα αποτελέσματα της εκπαίδευσης για το νευρωνικό δύκτιο που επέλεξα πιο πάνω: A graph showing a line

Description automatically generated with medium confidence

A graph with a blue line

Description automatically generated

A grid of black and white lines

Description automatically generated

**Συμπεράσματα**

Η επιτυχής εκπαίδευση του νευρωνικού δικτύου να προσεγγίσει την ορθή ταξινόμηση των γραμμάτων, αποδεικνύει την αποτελεσματικότητα του αλγορίθμου Kohonen και της κατάλληλης δομής δικτύου. Αποδεικτικέ πως χάρτης μεγέθους 50x50 είναι αρκετός για να λύσει το πρόβλημα με μεγάλη ακρίβεια. Η γραφική αναπαράσταση των αποτελεσμάτων ενισχύει αυτό το συμπέρασμα, αποτυπώνοντας τη βελτίωση της απόδοσης του δικτύου κατά την πρόοδο της εκπαίδευσης.

**Σημείωση**

Περαιτέρω τρεξίματα του δυκτίου για την διαπίστωση της τοπολογίας μπορούν να βρεθούν στο αρχείο Results and Graphs.xlsx

**Κλάσεις που χρησιμοποιούνται στον Κώδικα**

Στον παρεχόμενο κώδικα, πολλές προσαρμοσμένες κλάσεις χρησιμοποιούνται για τη δημιουργία και τη διαχείριση ενός νευρωνικού δικτύου. Παρακάτω, θα συζητήσουμε καθεμία από αυτές τις κλάσεις και τους ρόλους τους στην υλοποίηση του νευρωνικού δικτύου.

### 1. ****Neuron****

Η κλάση Neuron αντιπροσωπεύει έναν νευρώνα στο νευρωνικό δίκτυο. Οι νευρώνες είναι οι βασικές μονάδες στο δίκτυο, υπεύθυνες για την επεξεργασία των εισροών, τον υπολογισμό της εξόδου και τη διάδοση του σφάλματος κατά τη διάρκεια της εκπαίδευσης. Τα βασικά χαρακτηριστικά και μέθοδοι αυτής της κλάσης περιλαμβάνουν:

### Class Attributes:

### id (static): A static class-level attribute used to assign unique IDs to each neuron instance.

### Constructor - \_\_init\_\_(self, isBias=False, isOutput=False):

### Initializes a new instance of the Neuron class.

### Parameters:

### isBias: Boolean indicating if the neuron is a bias neuron.

### isOutput: Boolean indicating if the neuron is an output neuron.

### Instance attributes initialized:

### id: Unique identifier for the neuron instance.

### connectedFromNeurons: List of neurons from which this neuron receives input.

### connectedToNeurons: List of neurons to which this neuron sends output.

### output: The output value of the neuron (default is 1).

### delta: Used for storing the delta value during backpropagation.

### weights: List storing the weights of connections to this neuron.

### weightsOld: List to store previous weights (not used in the given code).

### isBias: Indicates if the neuron is a bias neuron.

### isOutput: Indicates if the neuron is an output neuron.

### Methods:

### addConnectedFromNeuron(neuron): Adds a neuron to the list of neurons from which this neuron receives input.

### addConnectedToNeuron(neuron): Adds a neuron to the list of neurons to which this neuron sends output.

### setConnectedToNeurons(neurons:list): Sets the list of neurons to which this neuron sends output.

### setOutput(output): Sets the output value of the neuron.

### getOutput(): Returns the output value of the neuron.

### setError(error): Sets the error value for the neuron.

### getError(): Returns the error value of the neuron.

### setDelta(delta): Sets the delta value (used in backpropagation).

### getDelta(): Returns the delta value.

### setWeights(weights): Sets the weights of the connections to this neuron.

### setWeight(i, j, weight): Sets a specific weight in the weights matrix.

### getWeights(): Returns the weights of the connections to this neuron.

### getConnectedFromNeurons(): Returns a list of neurons from which this neuron receives input.

### getConnectedToNeurons(): Returns a list of neurons to which this neuron sends output.

### \_\_str\_\_(): Returns a string representation of the neuron, including its ID, type (bias or output), connections, output, delta, and weights.

### 2. ****SOMNeuralNetwork****

### Η κλάση SOMNeuralNetwork αντιπροσωπεύει ένα νευρωνικό δίκτυο αυτο-οργανωτικών χαρτών (Self-Organizing Maps - SOM). Είναι σχεδιασμένη για την κατηγοριοποίηση και οπτικοποίηση δεδομένων, όπως γράμματα του αλφαβήτου. Οι βασικές λειτουργίες και χαρακτηριστικά περιλαμβάνουν:

### Attributes:

### parameters: Dictionary containing the parameters for the network.

### gridSize: Size of the grid.

### numEpochs: Number of epochs for training.

### learningRate: Initial learning rate.

### startLearningRate: Starting learning rate.

### numInputNeurons: Number of input neurons.

### gaussianRadius: Gaussian radius used in calculations.

### currentPattern: Current pattern being processed.

### currentEpoch: Current epoch in the training process.

### inputNeurons: List of input neurons.

### TrainingError: List to track training error.

### TestingError: List to track testing error.

### letterMap: Map of letters categorized by the SOM.

### Methods:

### appendTrainingError(trainingError): Appends a training error to the list.

### appendTestingError(testingError): Appends a testing error to the list.

### createInputNeurons(): Creates and initializes input neurons.

### initializeWeights(): Initializes weights for each input neuron.

### setInputs(inputs): Sets inputs for the input neurons.

### findBMU(): Finds and returns the Best Matching Unit (BMU).

### updateLearningRate(epoch=None): Updates the learning rate.

### updateGaussianRadius(epoch=None): Updates the Gaussian radius.

### updateWeights(bmu): Updates the weights of the input neurons based on the BMU.

### mapLetters(input\_data): Maps letters onto the grid based on input data.

### visualize\_map(): Visualizes the SOM map.

### visualize\_to\_txt(filename='clustering.txt'): Saves the visualization of the SOM map to a text file.

### plotTrainingError(): Plots the training error over epochs.

### plotTestingError(): Plots the testing error over epochs.

### getGrid(): Returns the grid of the SOM.

### getInputNeurons(): Returns the list of input neurons.

### getLearningRate(): Returns the current learning rate.

### save(filename): Saves the state of the SOMNeuralNetwork object to a file.

### \_\_str\_\_(): Returns a string representation of the SOMNeuralNetwork, detailing its parameters and nodes.

### 3. ****FileReader and FileWriter****

Αυτές οι κλάσεις χειρίζονται την ανάγνωση και εγγραφή σε αρχεία, που χρησιμοποιούνται ιδιαίτερα για την ανάγνωση δεδομένων εκπαίδευσης και δοκιμής, καθώς και παραμέτρων, και εγγραφή αρχείων καταγραφής σφαλμάτων και ποσοστού επιτυχίας.

**4. Utility**

Χρησιμοποιήθκηε για την οργάνωση και διαχώριση των δεδομένων