Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Τομέας Ηλεκτρονικής και Υπολογιστών

Artificial Neural Networks

1 Εισαγωγή

1.1 Εισαγωγή στον αλγόριθμο ANN

Ο αλγόριθμος perceptron είναι ένα είδος τεχνητού νευρωνικού δικτύου που προσεγγίζει γραμμικά προβλήματα 2 κλάσεων. Δεδομένων των εισόδων x και της εξόδου y, το perceptron εκπαιδεύεται υπολογίζοντας τα βάρη w που είναι τέτοια ώστε η παρακάτω συνάρτηση να είναι θετική για τη μία κλάση (κλάση C1) και αρνητική για την άλλη (κλάση C2):

$$d(x) = w^T x = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

Κατά την εκπαίδευση τα βάρη ανανεώνονται επαναληπτικά με βάση το σετ εκπαίδευσης. Ένα νευρωνικό δίκτυο (Artificial Neural Network – ANN) μπορεί να αποτελείται από περισσότερα από ένα perceptrons ενώ υπάρχουν διάφοροι τύποι εκπαίδευσης (π.χ. back propagation).

1.2 ANN στην Python

Για νευρωνικά δίκτυα στην Python, μπορούμε να χρησιμοποιήσουμε τη βιβλιοθήκη sklearn.neural_network:

```
>>> from sklearn.neural_network import MPLRegressor, MLPClassifier
```

Για να κατασκευάσουμε ένα μοντέλο τρέχουμε την εντολή `MPLRegressor`:

```
>>> model = MLPRegressor(hidden_layer_sizes=(100,)).fit(Xtrain, ytrain)
```

όπου με το hidden_layer_sizes επιλέγουμε τον αριθμό των στρωμάτων (layers) και των νευρώνων (neurons). Μπορούμε ακόμα να επιλέξουμε παραμέτρους όπως τον αλγόριθμο εκμάθησης (solver), τον ρυθμό εκμάθησης (learning_rate_init), τη συνάρτηση ενεργοποίησης, κτλ.

Έχοντας ένα μοντέλο, για να προβλέψουμε μια νέα τιμή τρέχουμε την εντολή

```
>>> clf.predict(xtest)
```

όπου επιστρέφεται μια τιμή ανάμεσα στις δύο κλάσεις.

2 Κατασκευή Μοντέλου Perceptron και ANN

Για την κατασκευή ενός μοντέλου ANN θα χρησιμοποιήσουμε ως εφαρμογή τα δεδομένα εκπαίδευσης του παρακάτω πίνακα για ένα πρόβλημα δυαδικής ταξινόμησης.

X1	X2	Y
0	0	1
0	1	1
1	0	-1
1	1	-1

Θα απαντήσουμε στα παρακάτω ερωτήματα:

α) Σχεδιάστε τα δεδομένα με διαφορετικά χρώματα/σύμβολα για κάθε κλάση.

β) Να εφαρμοστεί ο αλγόριθμος Perceptron για την εύρεση γραμμικής συνάρτησης απόφασης που διαχωρίζει τις δύο κλάσεις. Να χρησιμοποιηθεί w(1) = [0, 0, 0] και learning rate c = 1.

γ) Επαναλάβετε το ερώτημα (β) χρησιμοποιώντας την Python.

2.1 Κατασκευή Δεδομένων και Εισαγωγή Βιβλιοθηκών

Αρχικά κατασκευάζουμε τα δεδομένα και φορτώνουμε τις απαραίτητες βιβλιοθήκες:

```
>>> import pandas as pd

>>> import matplotlib.pyplot as plt

>>> anndata = pd.DataFrame({"X1": [0, 0, 1, 1], "X2": [0, 1, 0, 1], "Y":
[1, 1, -1, -1]})

>>> X = anndata.loc[:, ["X1", "X2"]]

>>> y = anndata.loc[:, "Y"]
```

2.2 Εφαρμογή Αλγορίθμου Perceptron

Για το ερώτημα (α) εκτελούμε:

```
>>> plt.scatter(anndata[(anndata.Y == -1)].X1, anndata[(anndata.Y ==
-1)].X2, c="red", marker="+")

>>> plt.scatter(anndata[(anndata.Y == 1)].X1, anndata[(anndata.Y ==
1)].X2, c="blue", marker="o")

>>> plt.show()
```

Για τα ερωτήματα (β), (γ), αρκεί να βρούμε τα βάρη του perceptron ώστε η παρακάτω συνάρτηση να είναι θετική για Y = 1 και αρνητική για Y = -1:

$$d(x) = w^T x = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

Αρχικά, τα βάρη είναι w1 = 0, w2 = 0, w3 = 0. Εκτελούμε επαναλήψεις και σε κάθε επανάληψη προσαρμόζουμε τα βάρη ανάλογα με το αν το σημείο θα έπρεπε να ανήκει στην κλάση 1 ή στην κλάση -1. Αν ένα σημείο x(i) ανήκει στην κλάση 1 ενώ θα έπρεπε να ανήκει στην -1, τότε προσθέτουμε στα βάρη τον πίνακα c·x(i). Αν ανήκει στην κλάση -1 ενώ θα έπρεπε να ανήκει στην 1, τότε αφαιρούμε από τα βάρη τον πίνακα c·x(i).

<u>1η επανάληψη</u>

1. $w^T(1)x(1) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0$, άρα $w(2) = w(1) + x(1) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

2. $w^T(2)x(2) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = 1 > 0$, άρα $w(3) = w(2) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

3. $w^T(3)x(3) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = 1 > 0$, άρα $w(4) = w(3) - x(3) = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}$

4. $w^T(4)x(4) = \begin{bmatrix} -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = -1 < 0$, άρα $w(5) = w(4) = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}$

<u>2η επανάληψη</u>

5. $w^T(5)x(1) = \begin{bmatrix} -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0$, άρα $w(6) = w(5) + x(1) = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$

6. $w^T(6)x(2) = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = 1 > 0$, άρα $w(7) = w(6) = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$

7. $w^T(7)x(3) = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = 0$, άρα $w(8) = w(7) - x(3) = \begin{bmatrix} -2 \\ 0 \\ 0 \end{bmatrix}$

8. $w^T(8)x(4) = \begin{bmatrix} -2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = -2 < 0$, άρα $w(9) = w(8) = \begin{bmatrix} -2 \\ 0 \\ 0 \end{bmatrix}$

<u>3η επανάληψη</u>

9. $w^T(9)x(1) = \begin{bmatrix} -2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0$, άρα $w(10) = w(9) + x(1) = \begin{bmatrix} -2 \\ 0 \\ 1 \end{bmatrix}$

10. $w^T(10)x(2) = \begin{bmatrix} -2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = 1 > 0$, άρα $w(11) = w(10) = \begin{bmatrix} -2 \\ 0 \\ 1 \end{bmatrix}$

11. $w^T(11)x(3) = [-2 \quad 0 \quad 1]\begin{bmatrix}1\\0\\1\end{bmatrix} = -1 < 0$, άρα $w(12) = w(11) = \begin{bmatrix}-2\\0\\1\end{bmatrix}$

12. $w^T(12)x(4) = [-2 \quad 0 \quad 1]\begin{bmatrix}1\\1\\1\end{bmatrix} = -1 < 0$, άρα $w(13) = w(12) = \begin{bmatrix}-2\\0\\1\end{bmatrix}$

<u>4η επανάληψη</u>

13. $w^T(13)x(1) = [-2 \quad 0 \quad 1]\begin{bmatrix}0\\0\\1\end{bmatrix} = 1 > 0$, άρα $w(14) = w(13) = \begin{bmatrix}-2\\0\\1\end{bmatrix}$

14. $w^T(14)x(2) = [-2 \quad 0 \quad 1]\begin{bmatrix}0\\1\\1\end{bmatrix} = 1 > 0$, άρα $w(15) = w(14) = \begin{bmatrix}-2\\0\\1\end{bmatrix}$

15. $w^T(15)x(3) = [-2 \quad 0 \quad 1]\begin{bmatrix}1\\0\\1\end{bmatrix} = -1 < 0$, άρα $w(16) = w(15) = \begin{bmatrix}-2\\0\\1\end{bmatrix}$

16. $w^T(16)x(4) = [-2 \quad 0 \quad 1]\begin{bmatrix}1\\1\\1\end{bmatrix} = -1 < 0$, άρα $w(17) = w(16) = \begin{bmatrix}-2\\0\\1\end{bmatrix}$

Ο αλγόριθμος έχει συγκλίνει διότι πλέον δεν συμβαίνει καμία διόρθωση στα βάρη. Η συνάρτηση απόφασης που προκύπτει είναι:

$d(x) = w^T x = [-2 \quad 0 \quad 1]\begin{bmatrix}x_1\\x_2\\1\end{bmatrix} = -2x_1 + 1$ Ή αλλιώς η ευθεία $x_1 = \frac{1}{2}$.

Επαναλαμβάνουμε τη λύση με την Python (ερώτημα (γ)) εκτελώντας τις παρακάτω εντολές:

```
>>> from sklearn.neural_network import MLPRegressor
>>> clf = MLPRegressor(hidden_layer_sizes=(), learning_rate_init=1)
>>> clf = clf.fit(X, y)
```

Μπορούμε επιπλέον να προβλέψουμε ένα νέο δείγμα με την παρακάτω εντολή:

```
>>> print(clf.predict([[0, 0]]))
>>> print(clf.predict([[1, 0]]))
>>> print(clf.predict([[0.5, 0]]))
```

4 Κατασκευή Μοντέλου ANN

Για την εκπαίδευση ενός νευρωνικού δικτύου (ANN) θα χρησιμοποιήσουμε ως εφαρμογή τα δεδομένα εκπαίδευσης του αρχείου που δίνεται (alldata.txt) για ένα πρόβλημα δυαδικής ταξινόμησης.

Ένα summary των δεδομένων είναι το παρακάτω:

```
      X1                    X2                  y
 Min.   :-3.25322007   Min.   :-4.664161   Min.   :1.0
 1st Qu.:-0.70900886   1st Qu.: 0.625361   1st Qu.:1.0
 Median :-0.01162501   Median : 1.641370   Median :1.5
 Mean   : 0.03493570   Mean   : 1.939284   Mean   :1.5
 3rd Qu.: 0.73337179   3rd Qu.: 3.115350   3rd Qu.:2.0
 Max.   : 3.63957363   Max.   : 9.784076   Max.   :2.0
```

Αρχικά, εισάγουμε τα δεδομένα και τα διαχωρίζουμε σε training set και test set:

```
>>> alldata = pd.read_csv("./alldata.txt")
>>> xtrain = alldata.loc[0:600, ["X1", "X2"]]
>>> ytrain = alldata.loc[0:600, "y"]
>>> xtest = alldata.loc[600:800, ["X1", "X2"]]
>>> ytest = alldata.loc[600:800, "y"]
```

Στη συνέχεια, θα απαντήσουμε στα παρακάτω ερωτήματα:

α) Σχεδιάστε τα training data με διαφορετικά χρώματα/σύμβολα για κάθε κλάση.

β) Κατασκευάστε ένα νευρωνικό δίκτυο με 2 hidden layers και 20 hidden nodes ανά layer και μέγιστο αριθμό επαναλήψεων 10.000 (max_iter = 10000).

γ) Υπολογίστε το σφάλμα εκπαίδευσης (training error) και το σφάλμα ελέγχου (testing error) στο training set και στο test set αντίστοιχα.

δ) Επαναλάβετε τα ερωτήματα (β) και (γ) για 3 hidden layers και 20 hidden nodes ανά layer και μέγιστο αριθμό επαναλήψεων 10.000 (max_iter = 10000). Τι παρατηρείτε;

4.1 Εισαγωγή Βιβλιοθηκών

Αρχικά φορτώνουμε τις απαραίτητες βιβλιοθήκες:

```
>>> import matplotlib.pyplot as plt

>>> import numpy as np
```

4.2 Κατασκευή ANN

Για το ερώτημα (α) εκτελούμε:

```
>>> plt.scatter(xtrain[(ytrain == 2)].X1, xtrain[(ytrain == 2)].X2,
c="red", marker="+")

>>> plt.scatter(xtrain[(ytrain == 1)].X1, xtrain[(ytrain == 1)].X2,
c="blue", marker="o")

>>> plt.show()
```

Για το ερώτημα (β) εκτελούμε:

```
>>> from sklearn.neural_network import MLPRegressor

>>> clf = MLPRegressor(hidden_layer_sizes=(2, 2), tol=0.01)

>>> clf = clf.fit(xtrain, ytrain)
```

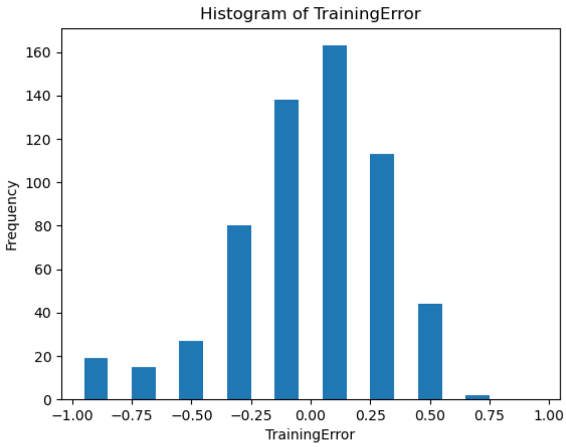
Για το ερώτημα (γ) έχουμε:

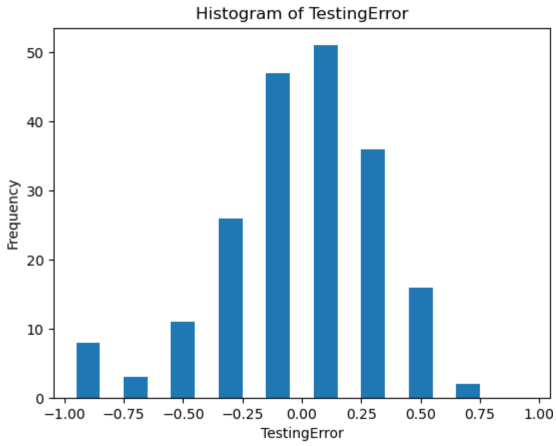
```
## Training Error

>>> pred = clf.predict(xtrain)
>>> trainingError = [(t - p) for (t, p) in zip(ytrain, pred)]
>>> MAE = np.mean(np.abs(trainingError))
>>> print(MAE)
>>> plt.hist(trainingError, range=(-1, 1), rwidth=0.5)
>>> plt.show()

## Testing Error

>>> pred = clf.predict(xtest)
>>> testingError = [(t - p) for (t, p) in zip(ytest, pred)]
>>> MAE = np.mean(np.abs(testingError))
>>> print(MAE)
>>> plt.hist(testingError, range=(-1, 1), rwidth=0.5)
>>> plt.show()
```


Histogram of TrainingError
Frequency
TrainingError
0
20
40
60
80
100
120
140
160
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Histogram of TestingError
Frequency
TestingError
0
10
20
30
40
50
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00