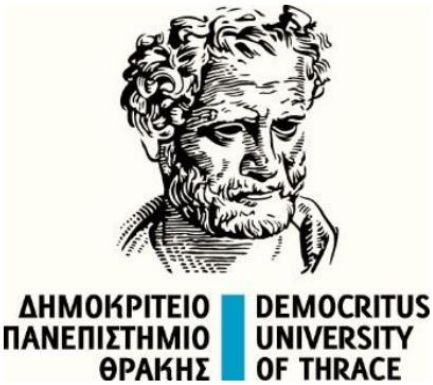


7Η ΕΡΓΑΣΙΑ

Στο μάθημα:
«Αναγνώριση Προτύπων»
Καθηγητής: Νικόλαος Μητιανούδης

Στυλιανός Μούσλεχ
ΑΜ:57382
20/12/2020 ,Ξάνθη

ΕΙΣΑΓΩΓΗ

Η υλοποίηση γίνεται στην κάνοντας χρήση της βιβλιοθήκης scikit-learn. Η λογική του Validation που ακολούθησα για να έχει περισσότερο νόημα είναι η k-folds cross validation με τον ίδιο τρόπο που αναλύθηκε και στην προηγουμένη εργασία για να έχουμε και συγκρίσιμα αποτελέσματα

ΆΣΚΗΣΗ 7.1: ΓΡΑΜΜΙΚΟ PERCEPTRON

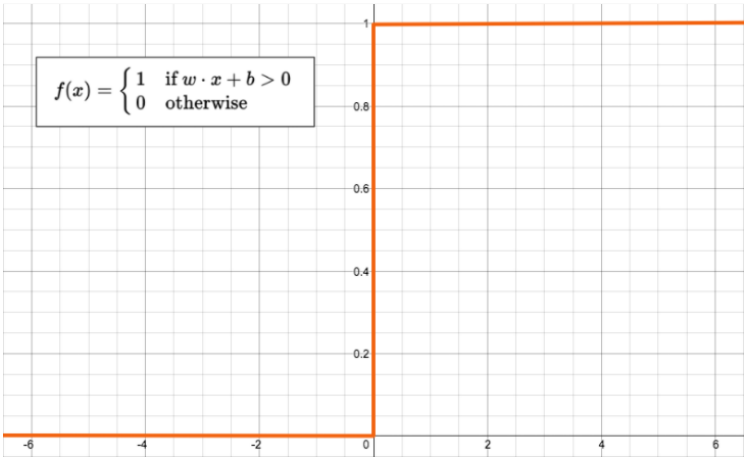
Ο κώδικας του ερωτήματος γίνεται στο αρχείο “Ex7_1.py”.

Όπως βλέπουμε τα δεδομένα που μας δίνονται είναι γραμμικά διαχωρίσιμα και περιμένουμε το perceptron να τα χωρίσει σωστά με μια ευθεία.

Αρχικά λόγω της απλότητας και του μεγέθους του προβλήματος μπορούμε να κάνουμε μια αναλυτική μαθηματική λύση.

Αναλυτική Λύση

Σε πρόβλημα classification 2 κλάσεων θα κάνουμε χρήση της συνάρτησης step function:



Έχουμε 2 χαρακτηριστικά άρα έχουμε 2 βάρη w1 και w2 καθώς και ένα bias w0

Όταν ένα δείγμα ανήκει στην κλάση ω1 τότε θα έχουμε έξοδο στο perceptron 0 και όταν στην ω2 θα έχουμε έξοδο 1, Άρα μαθηματικά

$$y = w_0 + \sum_{i=1}^{2} w_i \gtrless_{\omega_1}^{\omega_2} 0$$

Η αναβάθμιση των βαρών θα γίνει με βάση τον κανόνα Perceptron:

$$w_i(t+1) = w_i(t) \; + \; learningRate{*}(trueLabel - predictedLabel){*}x_i \; , i = 1,2$$
$$w_0(t+1) = w_0(t) \; + \; learningRate{*}(trueLabel - predictedLabel)$$

Στην περίπτωση μας βάζουμε learningRate=1.

Ξεκινάμε αρχικοποιώντας όλα τα βάρη στη τιμή 1 και κάνουμε τη διαδικασία της μάθησης με online τρόπο , δηλαδή μετά από κάθε σημείο ανανεώνουμε τα βάρη:

$x = (-2, 1):$
$y = 1 - 2 + 1 = 0$ άρα *prediction* ω_1 ενώ ανήκει ω_2
$w_{new1} = 1 + 1*(1 - 0)*(-2) = -1$
$w_{new2} = 1 + 1*(1 - 0)*(1) = 2$
$w_{new0} = 1 + 1 = 2$

$x = (1, 2):$
$y = 2 - 1 + 4 = 6 \rightarrow 1$ άρα *prediction* ω_2 και ανήκει ω_2
Βάρη μένουν ίδια

$x = (0, 0):$
$y = 2 + 0 + 0 = 2 \rightarrow 1$ άρα *prediction* ω_2 ενώ ανήκει ω_1
$w_{new1} = -1 + 1*(0 - 1)*(0) = -1$
$w_{new2} = 2 + 1*(0 - 1)*(0) = 2$
$w_{new0} = 2 - 1 = 1$

$x = (-2, -1):$
$y = 1 + 2 - 2 = 1 \rightarrow 1$ άρα *prediction* ω_2 ενώ ανήκει ω_1
$w_{new1} = -1 + 1*(0 - 1)*(-2) = +1$
$w_{new2} = 2 + 1*(0 - 1)*(-1) = +3$
$w_{new0} = 1 - 1 = 0$

$x = (2, 0):$
$y = 0 + 2 + 0 = 2 \rightarrow 1$ άρα *prediction* ω_2 ενώ ανήκει ω_1
$w_{new1} = +1 + 1*(0 - 1)*(2) = -1$
$w_{new2} = 3 + 1*(0 - 1)*(0) = +3$
$w_{new0} = 0 - 1 = -1$

$x = (-2, 1):$
$y = -1 + 2 + 3 = 5 \rightarrow 1$ άρα *prediction* ω_2 και ανήκει ω_2
Τα βάρη μένουν ίδια

$x = (2, 1):$
$y = -1 - 2 + 3 = 0 \rightarrow 0$ άρα *prediction* ω_1 και ανήκει ω_1
Τα βάρη μένουν ίδια

$x = (1, 2):$
$y = -1 - 1 + 6 = 4 \rightarrow 1$ άρα *prediction* ω_2 και ανήκει ω_2
Τα βάρη μένουν ίδια

$x = (0, 0)$:
$y = -1 + 0 = -1 \rightarrow 0$ άρα *prediction* ω_1 και ανήκει ω_1
Τα βάρη μένουν ίδια

$x = (-2, -1)$:
$y = -1 + 2 - 3 = -2 \rightarrow 0$ άρα *prediction* ω_1 και ανήκει ω_1
Τα βάρη μένουν ίδια

$x = (2, 0)$:
$y = -1 - 2 + 0 = -3 \rightarrow 0$ άρα *prediction* ω_1 και ανήκει ω_1
Τα βάρη μένουν ίδια

Φτάσαμε στο σημείο όπου όλα τα σημεία ταξινομούνται σωστά και δεν κάνουμε αλλαγές στα βάρη οπότε η εκπαίδευση έχει τελειώσει με τα παρακάτω βάρη:

$$w = [-1, +3] \, , w_0(bias) = [-1]$$

Μπορούμε να κάνουμε plot την συνάρτηση απόφασης μας με βάση τον τύπο:

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

Αυτό θα κάνουμε παρακάτω ενώ θα κάνουμε και την ίδια διαδικασία με την συνάρτηση της scikit learn για να δούμε και μια διαφορετική λύση

Άρα την ίδια διαδικασία μπορούμε να την κάνουμε με την συνάρτηση perceptron :

```
X=np.array([[-2,1],[1,2],[0,0],[-2,-1],[2,0],[2,1]])
Y=np.array([1,1,0,0,0,0])

model = Perceptron(tol=1e-3, random_state=0)
model.fit(X,Y)
print('weights: ',model.coef_,"and bias",model.intercept_)
print("Single Perceptron Accuracy",model.score(X,Y))
```

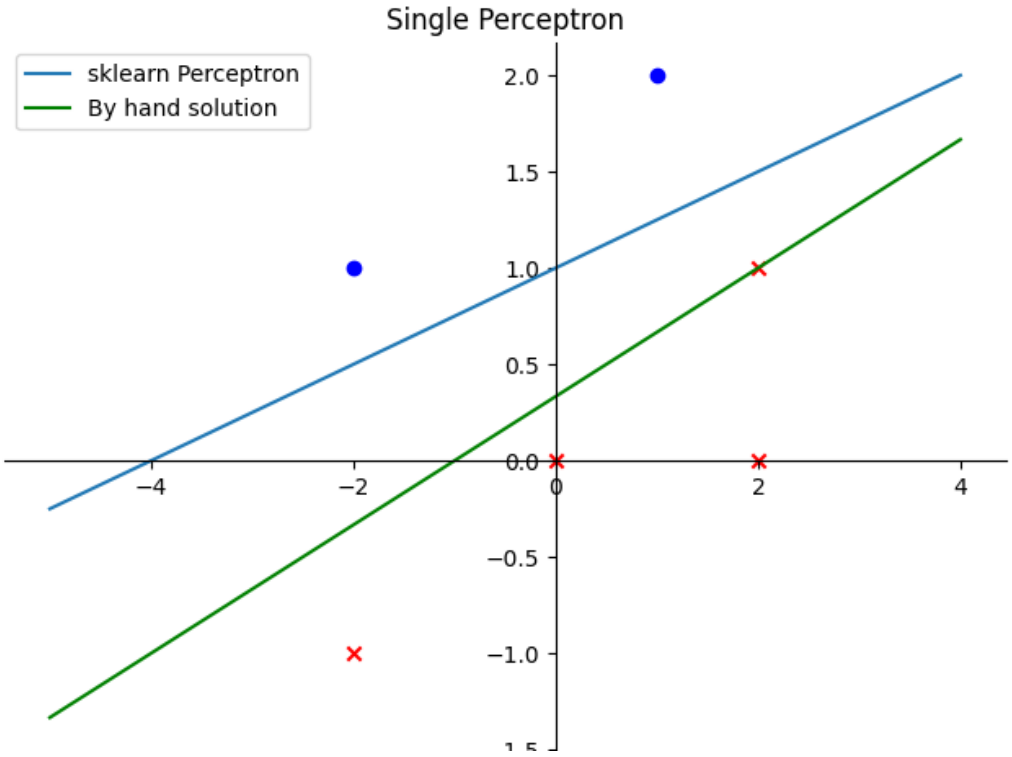
Και έχουμε τα παρακάτω αποτελέσματα:

```
weights:  [[-1.  4.]] and bias [-4.]
Single Perceptron Accuracy 1.0
```

Και μπορούμε να κάνουμε plot την ευθεία απόφασης με βάσει τον τύπο που είπαμε πριν:

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

Μαζί θα βάλουμε και με βάση τα βάρη που υπολογίσαμε με την λύση στο χέρι:



Παρατηρούμε ότι και οι 2 λύσεις είναι σωστές, οι διαφορές εδώ προκύπτουν λόγω διαφορετικών αρχικοποιήσεων

ΆΣΚΗΣΗ 7.2: IRIS Data Set με Ταξινομητή Πολυστρωματικά Perceptron

Α.Ο κώδικας βρίσκεται στο αρχείο Ex7_2singleLayer.py”

Η συγκεκριμένη αρχιτεκτονική ενός στρώματος perceptron που αποτελείται από αριθμό νευρώνων όσων είναι και οι κλάσεις , γίνεται απευθείας από την συνάρτηση Perceptron καθώς βλέπει τον συνολικό αριθμό τιμών στα labels και παράγει τόσα perceptrons. Επίσης η ασχοκοποίηση σε τυχαίες τιμές γίνεται αυτόματα από τη συνάρτηση και δεν χρειάζεται να κάνουμε αρχικοποιήσεις.

Οπότε με βάση την λογική του train/validation/test που είπαμε στην εισαγωγή εκπαιδεύουμε το μοντέλο:

```
iris=load_iris()
X=iris.data
Y=iris.target

print("Single Layer Perceptron - One vs All ")
X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size=0.2, random_state=0,stratify=Y)
model = Perceptron(tol=1e-3,n_jobs=1)
model.fit(X_train,y_train)
predict_train = model.predict(X_train)
```

```
print("Train accuracy:", accuracy_score(predict_train, y_train))
scores = cross_val_score(model,X_train, y_train, cv=4)
print("Validation Accuracy ",": %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
predict=model.predict(X_test)
print("Test accuracy:",accuracy_score(y_test,predict))
```

και έχουμε τα παρακάτω αποτελέσματα:

```
Single Layer Perceptron - One vs All
Train accuracy: 0.9416666666666667
Validation Accuracy  : 0.78 (+/- 0.23)
Test accuracy: 0.9666666666666667
```

Εδώ βλέπουμε ότι στο test accuracy έχει πολύ καλό σκορ, αυτό βέβαια ειδικά στην περίπτωση μας που έχουμε πολύ μικρό dataset και ακόμα μικρότερο test set το accuracy δεν μας δίνει πάντα αντιπροσωπευτική πληροφορία, σε ένα άλλο μπορεί να είχαμε πολύ μικρότερο. Μπορούμε να δούμε πολύ καλύτερα πως πηγαίνει το μοντέλο βλέποντας το μέσο validation accuracy και την απόκλιση που υπάρχει, βλέπουμε λοιπόν ότι υπάρχει μεγάλη απόκλιση εκεί. (Μου φαίνεται λογικό το αποτέλεσμα καθώς ο γραμμικός διαχωρισμός είναι εύκολος μεταξύ κλάσης 0 και 1,2 αλλα όχι μεταξύ 1 και 2 πάντα, άρα αν τα validation, ή και test data περιέχουν πολλά δεδομένα της 0 θα είναι πιο εύκολο από κάποια άλλα).

Β. Ο κώδικας βρίσκεται στο αρχείο Ex7_2mlp.py”

Εδώ πλέον θα κάνουμε πολυστρωματικά perceptron, το input και output layer αναγνωρίζονται αυτόματα στην sklearn οπότε εμείς ορίζουμε το πλήθος και μέγεθος των hidden layers, καθώς και άλλες υπερπαραμέτρους,

Πιο συγκεκριμένα την μέθοδο εκμάθησης όπου διαλέγουμε Stochastic Gradient Descend που κάνει χρήση του Backpropagation για τον υπολογισμό των gradients

Επίσης το learning rate, εκεί γίναν αρκετές δοκιμές το αποτέλεσμα ήταν αναμενόμενο, με μικρό learning rate δεν έφτανε στο max iteration σε κάποια καλή σύγκλιση με πολύ μεγάλο δεν είχαμε καλή σύγκλιση για τους λόγους που έχουμε δει και στην πρώτη εργασία με το gradient descend.

Τέλος δοκιμάστηκαν και διάφορες activation functions για να συγκρίνουμε αποτελέσματα.

Συγκεντρωτικά όλα τα αποτελέσματα:

```
                          Multi Layer Perceptron - One hidden layer
========================================================================
Number of Neurons in Hidden Layer= 2
Activation Function:logistic
Train accuracy: 0.85
Validation Accuracy  : 0.90 (+/- 0.07)
Test accuracy: 0.8666666666666667
========================================================================
Number of Neurons in Hidden Layer= 5
Activation Function:logistic
Train accuracy: 0.975
Validation Accuracy  : 0.97 (+/- 0.05)
Test accuracy: 1.0
========================================================================
Number of Neurons in Hidden Layer= 10
Activation Function:logistic
Train accuracy: 0.975
Validation Accuracy  : 0.98 (+/- 0.06)
Test accuracy: 1.0
========================================================================
Number of Neurons in Hidden Layer= 50
Activation Function:logistic
Train accuracy: 0.975
Validation Accuracy  : 0.98 (+/- 0.06)
Test accuracy: 1.0
========================================================================
```

```
========================================
Number of Neurons in Hidden Layer= 2
Activation Function:relu
Train accuracy: 0.975
Validation Accuracy  : 0.97 (+/- 0.07)
Test accuracy: 1.0
========================================
Number of Neurons in Hidden Layer= 5
Activation Function:relu
Train accuracy: 0.975
Validation Accuracy  : 0.97 (+/- 0.05)
Test accuracy: 1.0
========================================
Number of Neurons in Hidden Layer= 10
Activation Function:relu
Train accuracy: 0.975
Validation Accuracy  : 0.97 (+/- 0.05)
Test accuracy: 1.0
========================================
Number of Neurons in Hidden Layer= 50
Activation Function:relu
Train accuracy: 0.975
Validation Accuracy  : 0.97 (+/- 0.05)
Test accuracy: 1.0
========================================
```

```
========================================
Number of Neurons in Hidden Layer= 2
Activation Function:tanh
Train accuracy: 0.9583333333333334
Validation Accuracy  : 0.94 (+/- 0.07)
Test accuracy: 1.0
========================================
Number of Neurons in Hidden Layer= 5
Activation Function:tanh
Train accuracy: 0.95
Validation Accuracy  : 0.95 (+/- 0.06)
Test accuracy: 1.0
========================================
Number of Neurons in Hidden Layer= 10
Activation Function:tanh
Train accuracy: 0.975
Validation Accuracy  : 0.97 (+/- 0.05)
Test accuracy: 1.0
========================================
Number of Neurons in Hidden Layer= 50
Activation Function:tanh
Train accuracy: 0.9833333333333333
Validation Accuracy  : 0.97 (+/- 0.05)
Test accuracy: 1.0
```

Γενικά συμπεράσματα που βγάζουμε από τα αποτελέσματα είναι ότι γενικότερα το να έχουμε ένα hidden layer είναι πολύ καλύτερο από το να έχουμε μόνο 1 layer. Επίσης τα αποτελέσματα είναι πάρα πολύ καλά και μάλιστα λίγο καλύτερα από τα SVM της προηγούμενης εργασίας αλλά όχι με τεράστια διαφορά. Τέλος βλέπουμε ότι και με περισσότερους νευρώνες έχουμε λίγη βελτίωση φυσικά δεν φαίνεται πολύ καθώς εξαρχής έχουμε πολύ καλά αποτελέσματα.

Γ. Επαναλαμβάνουμε την ίδια διαδικασία με 2 hidden layers και βάζουμε τον ίδιο αριθμό νευρώνων ανά layer:

```
                    Multi Layer Perceptron - Two hidden layer
================================================================================================
Number of Neurons in Hidden Layer= 2
Activation Function:logistic
Train accuracy: 0.3333333333333333
Validation Accuracy  : 0.33 (+/- 0.03)
Test accuracy: 0.3333333333333333
================================================================================================
Number of Neurons in Hidden Layer= 5
Activation Function:logistic
Train accuracy: 0.3416666666666667
Validation Accuracy  : 0.34 (+/- 0.03)
Test accuracy: 0.3333333333333333
================================================================================================
Number of Neurons in Hidden Layer= 10
Activation Function:logistic
Train accuracy: 0.3333333333333333
Validation Accuracy  : 0.33 (+/- 0.00)
Test accuracy: 0.3333333333333333
================================================================================================
Number of Neurons in Hidden Layer= 50
Activation Function:logistic
Train accuracy: 0.9833333333333333
Validation Accuracy  : 0.98 (+/- 0.06)
Test accuracy: 1.0
================================================================================================
```

```
================================================
Number of Neurons in Each Hidden Layer= 2
Activation Function:relu
Train accuracy: 0.9583333333333334
Validation Accuracy  : 0.95 (+/- 0.06)
Test accuracy: 1.0
================================================
Number of Neurons in Each Hidden Layer= 5
Activation Function:relu
Train accuracy: 0.6583333333333333
Validation Accuracy  : 0.65 (+/- 0.03)
Test accuracy: 0.6666666666666666
================================================
Number of Neurons in Each Hidden Layer= 10
Activation Function:relu
Train accuracy: 0.975
Validation Accuracy  : 0.95 (+/- 0.03)
Test accuracy: 1.0
================================================
Number of Neurons in Each Hidden Layer= 50
Activation Function:relu
Train accuracy: 0.975
Validation Accuracy  : 0.97 (+/- 0.05)
Test accuracy: 1.0
================================================
```

```
================================================
Number of Neurons in Each Hidden Layer= 2
Activation Function:tanh
Train accuracy: 0.39166666666666666
Validation Accuracy  : 0.39 (+/- 0.03)
Test accuracy: 0.4
================================================
Number of Neurons in Each Hidden Layer= 5
Activation Function:tanh
Train accuracy: 0.6416666666666667
Validation Accuracy  : 0.81 (+/- 0.26)
Test accuracy: 0.6666666666666666
================================================
Number of Neurons in Each Hidden Layer= 10
Activation Function:tanh
Train accuracy: 0.975
Validation Accuracy  : 0.97 (+/- 0.05)
Test accuracy: 1.0
================================================
Number of Neurons in Each Hidden Layer= 50
Activation Function:tanh
Train accuracy: 0.975
Validation Accuracy  : 0.97 (+/- 0.05)
Test accuracy: 1.0
```

Αυτή τη φορά παρατηρούμε ότι για να φτάσουμε σε εξίσου καλά αποτελέσματα πρέπει να έχουμε αρκετούς νευρώνες ανά layer καθώς με 2 και 5 και δεν έχουμε τόσο καλό αποτέλεσμα συγκριτικά με το 1 hidden layer. Τέλος υπάρχει μεγαλύτερη διαφοροποίηση μεταξύ activation functions σε σχέση με το 1 hidden layer.

Αυτά τα αποτελέσματα δεν μπορούμε βέβαια να πούμε ότι «γενικοποιούνται» πάντα σχετικά με την αρχιτεκτονική και τις άλλες υπερπαραμέτρους του Multi-Layer perceptron καθώς αυτό εξαρτάται από το είδος και το μέγεθος του dataset που έχουμε.