RETE NEURALE

Multilayer Perceptron

Il Multilayer Perceptron è un modello di rete neurale artificiale che mappa insiemi di dati in ingresso in un insieme di dati in uscita appropriati.

In questo Progetto è stata utilizzata Scikit-learn, una libreria open-source di apprendimento automatico (Libreria Python). È stata utilizzata la Classe MLPClassifier che implementa un algoritmo Multilayer Perceptron che apprende utilizzando l'algoritmo Backpropagation. Questo algoritmo è un metodo che serve ad allenare reti neurali artificiali.

Per utilizzarla, all'inizio del codice viene importata la libreria in questo modo.

from sklearn.neural_network import MLPClassifier

Implementazione del codice

All'inizio vengono importate tutte le librerie necessarie al fine di eseguire correttamente il codice.

```
import time
import numpy as np
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import KFold
from sklearn.metrics import classification_report
```

```
#Prendo il tempo iniziale del programma
tempo_iniziale = time.time()
#Leggiamo il file contenente il training set con as_matrix() lo convertiamo in Numpy-Array
training_set= pd.read_csv("lettere.txt", header= None).as_matrix()

#leggiamo la prima riga
riga= training_set[:, 1:]
colonna= training_set[:, 0]
kf = KFold(n_splits = 10, shuffle = True)

for i,z in kf.split(riga):
    neural_network= MLPClassifier(hidden_layer_sizes=(100),activation='logistic', max_iter= 500)
    #Adatta il modello alla matrice dati X e all'obiettivo y associ
    neural_network.fit(riga[i], colonna[i])

    predizione = neural_network.predict(riga[z])

    print(classification_report(colonna[z],predizione))


#Qui prendo il tempo finale del programma
tempo_finale = time.time()
#Calcolo il tempo di esecuzione del programma
calcolo_tempo = tempo_finale - tempo_iniziale
print ('Impiegati', calcolo_tempo, 'secondi.')
```

Per vedere quanto tempo ci mette il programma a esaminare un file con 20000 elementi è stato preso il tempo iniziale e finale. Preso il tempo iniziale il programma legge il file contenente il Training_set, all'interno del quale ci sono le lettere e le loro possibili rappresentazioni ad esempio:

```
T,6,8,6,6,3,4,12,3,7,12,10,5,1,11,1,5
G,2,3,2,2,1,6,8,5,4,9,8,9,2,8,4,9
E,5,10,4,5,3,7,8,4,3,10,5,9,4,9,7,10
J,6,7,8,8,7,8,8,5,7,7,5,8,4,10,10,8
Q,4,3,5,5,2,8,9,7,6,6,8,9,3,8,5,9
N,7,14,9,8,5,3,9,4,4,13,12,10,6,9,0,8
T,4,5,4,4,2,6,11,2,7,11,9,4,1,11,2,4
M,5,8,7,6,7,8,6,6,5,7,7,10,11,6,2,9
O,4,5,5,4,3,7,7,8,5,7,6,8,2,8,3,8
A,6,10,7,6,3,12,0,4,1,11,4,12,4,4,3,11
T,2,9,4,7,2,7,13,0,5,7,10,8,0,8,0,8
N,3,5,6,3,2,7,9,2,5,10,6,6,5,8,1,7
F,4,10,4,7,3,1,12,4,4,11,10,8,0,8,2,6
P,2,5,3,7,5,8,6,5,1,7,6,7,6,8,5,9
U,3,7,5,5,3,4,8,6,6,7,9,10,3,9,1,8
H,5,7,8,5,5,9,7,3,6,10,4,7,5,8,4,9
O,6,9,4,4,2,8,7,5,5,8,4,7,4,9,5,8
S,2,8,3,6,2,8,8,6,9,5,6,7,0,8,9,8
```

Attraverso la funzione KFold diciamo al programma quanti campioni alla volta deve esaminare … in questo caso è stato messo 10 come da specifica. In fine il programma stampa un report di testo che mostra le principali metriche di classificazione:

	precision	recall	f1-score	support
A	0.94	0.82	0.87	71
B	0.69	0.78	0.73	81
C	0.94	0.78	0.85	85
D	0.72	0.88	0.79	73
E	0.72	0.69	0.70	74
F	0.89	0.87	0.88	71
G	0.62	0.77	0.69	70
H	0.76	0.68	0.72	77
I	0.91	0.91	0.91	69
J	0.91	0.82	0.86	77
K	0.82	0.82	0.82	71
L	0.96	0.90	0.93	87
M	0.96	0.94	0.95	85
N	0.89	0.84	0.86	80
O	0.74	0.78	0.76	79
P	0.86	0.86	0.86	79
Q	0.78	0.74	0.76	73
R	0.76	0.77	0.76	88

R	0.76	0.77	0.76	88
S	0.60	0.68	0.64	60
T	0.88	0.78	0.83	78
U	0.84	0.85	0.84	78
V	0.93	0.92	0.93	88
W	0.83	0.95	0.88	75
X	0.84	0.82	0.83	74
Y	0.83	0.90	0.86	94
Z	0.84	0.76	0.80	63
avg / total	0.83	0.82	0.82	2000

Ed in fine il tempo impiegato a esaminare l'intero training set:

```
Impiegati 316.13402795791626 secondi.
```