Progetto MOBD: Classificatore SVM per Speller P300

Luglio 2019

Lanciotti Marco-Nedia Salvatore

***Sommario*—Nell'ambito del corso di Metodi di Ottimizzazione per Big Data (MOBD), anno accademico 2018-2019, è stato realizzato il progetto P300 Speller, il cui scopo è quello di creare un classificatore, tramite l'utilizzo di un SVM(support vector machine) che sia in grado di predire i caratteri osservati dal paziente soggetto a SLA e che sta utilizzando il tool BCI (Brain-Computer Interface)**

I. Descrizione del Problema

Lo scopo dell'utente è di copiare e scrivere 6 parole di cinque caratteri ciascuna. Per ciascuna parola, l'utente,tramite l'utilizzo di una matrice 6X6, è in grado di identificare i caratteri corretti componenti la parola da comunicare. Al fine di ottenere tale obiettivo viene sfruttata tale matrice in cui, in modo alternato, vengono illuminate tutte le varie righe e colonne per ogni carattere ad una frequenza di 4 Hz. Tuttavia una volta illuminate casualmente, solo una riga e una colonna saranno quelle effettivamente target. Si è osservato come le risposte evocate da questi rari stimoli (ad es. i 2 stimoli su 12 che contenevano il carattere desiderato) sono diversi da quelli evocati dagli stimoli che non contenevano il carattere desiderato. Ovviamente il carattere prescelto, a questo punto, è proprio quello ottenuto tramite l'intersezione della riga e colonna selezionate.

II. Introduzione

Per generare il modello capace di individuare la riga/colonna target, ovvero in cui è presente la P300, è stato prima scelto il tipo di classificatore e successivamente gli altri iperparametri. Dato che l'osservazione del carattere è ripetuta per 10 iterazioni sono stati effettuati due approcci. Nel primo approccio è stata effettua una media tra le 10 iterazioni, generando una iterazione media per ogni carattere e riducendo notevolmente il dataset. Nel secondo approccio è stato invece scelto di utilizzare tutto il dataset completo con le 10 iterazioni, ma effettuare dopo l'addestramento un operazione di max tra i valori predetti dal classificatore svm. Abbiamo notato che con entrambi gli approcci si ottengono livelli di accuratezza abbastanza simili tra di loro.

III. Data Preparation

La prima operazione che è stata effettuata è stato lo splitting dei dati in training test e test set. Si è deciso di utilizzare una politica 70/30, ovvero 70% nel training e 30% nel test, che corrispondono a esattemente 21 caratteri nel training e 9 nel test con le rispettive iterazioni. La decisione dello split è stata inizialmente di 80/20, tuttavia ci siamo resi conto che, in questo modo si avevano nella fase di cross validation meno folds, il che avrebbe portato conseguentemente meno iterazioni, per questo siamo passati alla configurazione finale di 70/30.

IV. Preprocessamento dei dati

Dopo aver effettuato la fase di DATA PREPARATION, in cui in sostanza abbiamo deciso le politiche con cui splittare il nostro dataset, ci siamo preoccupati di esaminare e trattare in modo accurato i dati da dare in pasto alla macchina da addestrare. Al fine di realizzare ciò, abbiamo deciso di ordinare i nostri dati per carattere, così da tenere traccia in modo semplice di tutte le iterazioni per ogni carattere. Infatti facendo cosi siamo stati in grado di capire che ogni blocco da 120 ordinato era riferito ad un dato carattere(12 righe/colone * 10 iterazioni). Ovviamente anche in questo caso abbiamo discriminato la casistica in cui stessimo utilizzando l'approccio mediato o su tutto il dataset ottenendo in entrambi i casi lo stesso risultato. Successivamente, ordinati i caratteri, abbiamo optato per la scelta della normalizzazione del dataset che ci ha aiutato nella separazione in vettori/matrici separate dei dati di interesse. Infatti sfruttando la normalizzazione abbiamo ottenuto 3 differenti matrici/vettori: una per i dati di train, gli stimoli effettivi, un vettore tenente traccia dell'ordinamento dei caratteri (vettore C) e un vettore di label contenente i target (vettore Y). Nonostante la normalizzazione, per effettuare le nostre prove, abbiamo deciso di tenere anche un dataset non normalizzato con il quale non abbiamo notato effetive differenze in termini prestazionali.

V. K-Cross Validation

Scelte tutte le politiche di gestione dei dati, abbiamo deciso di addestrare la nostra macchina al fine di raggiungere due decisioni imprescindibili:

- Scegliere il miglior classificatore da usare
- Scegliere il parametro C, ossia il giusto compromesso tra la minimizzazione dell'errore sul training set e la massimizzazione del margine.

Al fine di addestrare la nostra macchina in maniera efficiente abbiamo deciso di partizionare il nostro dataset in 7 fold da 3 caratteri ciascuno.
Per quanto riguarda la scelta del classificatore fin da subito abbiamo notato le migliori prestazioni ottenute tramite l'SVM lineare sia in termini di stabilità sia in termini di maggior semplicità nella praticità di utilizzo. Infatti è stato più comodo per noi lavorare su un kernel lineare che ci ha permesso di impostare in maniera efficiente una funzione max rispetto alla funzione segno che ci ha aiutato notevolmente nella predizione dei target corretti. Per quanto riguarda gli altri due classificatori abbiamo deciso di utilizzare l'SVM con funzione

segno che però, dava risultati discordanti. Infatti abbiamo notato l'incapacità della macchina SVM, in particolare nel caso polinomiale, di predire in maniera efficiente i decision value e in alcune casistiche andando in estremo over-fitting con molti più target sul singolo carattere dei 2 stabiliti.
Invece per quanto riguarda il parametro C anche in questo caso abbiamo addestrato su 7 fold da 3 caratteri e abbiamo testato la nostra macchina su 5 parametri differenti:
I) c = 10 II) c = 1 III) c = 0.1 IV) c = 100 V) c = 0.001
Abbiamo notato che i risultati migliori nel caso in cui abbiamo lavorato su tutto il dataset sono stati ottenuti tramite il parametro c = 0.001 mentre, nel caso mediato, il parametro c = 0.001 non risultava ottimo,anzi, in molte casisistiche risultava essere il peggiore; questo a vantaggio, in generale, del parametro c = 1. Al termine di questa fase di settaggio ci siamo fatti restituire una media, sui 7 fold, delle accurattezze ottenute tramite i diversi parametri e proprio sfruttando queste valutazioni abbiamo scelto tutti i paramentri migliori per il nostro modello di addestramento finale.

VI. Scelta del modello ottimo

Analizziamo in questa sezione i valori ottenuti per i due approcci nella crsoss validation.

1a. Nel caso dell'approccio con iterazione media i valori del parametro C sono quelli indicati in tabella:

Parametro C	AccuracyMeanN
10	0.8809524
1	0.8809524
0,1	0.9047619
100	0.8809524
0,001	0.8571429

La accuratezza ottenuta è un'accuracy media che è stata calcolata nel caso normalizzato con uno split in 7 fold di 3 caratteri ciascuno nel validation e i restanti 18 nel training.

Parametro C	AccuracyMean	AccuracyMean
10	1.0000000	0.9750000
1	1.0000000	0.9750000
0,1	0.9761905	0.9750000
100	1.0000000	0.9750000
0,001	0.8571429	0.875000

1b. In questa tabella,invece, esaminiamo i risultati ottenuti su un dataset non normalizzato costituito nel primo caso da 7 fold di 3 caratteri ciascuno mentre nel secondo caso da 5 fold (approssimati tramite utilizzo funzione floor()) di 4 caratteri ciascuno

Parametro C	AccuracyFullN
10	0.9285714
1	0.9047619
0,1	0.9523810
100	0.9285714
0,001	0.9761905

2.a Ora, in questa tabella, andiamo ad analizzare le accuracy ottenute tramite l'approccio "full" su un dataset normalizzato. In questa casistica l'accuracy media,come in precedenza, sarà sempre calcolata sulle 7 fold di 3 caratteri. In questa situazione, come si evince dalla tabella, il parametro c migliore risulta essere, come ci si aspettava, c = 0.001.

Parametro C	AccuracyFull	AccuracyFull)
10	0.9047619	0.9000000
1	0.9047619	0.8750000
0,1	0.9523810	0.9250000
100	0.9523810	0.8750000
0,001	0.9523810	0.9750000

2.b Infine andiamo ad analizzare le accuracy ottenute tramite l'approccio "full" su un dataset non normalizzato. In questa casistica l'accuracy media sarà calcolata sulle 7 fold di 3 caratteri ciascuna nel primo caso, mentre nel secondo caso da 5 fold (approssimati tramite utilizzo funzione floor()) di 4 caratteri ciascuno. Anche in questo caso il parametro c migliore risulta essere, come ci si aspettava, c = 0.001.

Conclusioni

Dall'analisi vista nella sezione precedente si può evincere come entrambi gli approcci utilizzati e implementati siano in realtà equamente efficienti. Tuttavia si è notato come l'approccio utilizzante la media sia meno stabile ed efficiente. Questo sopratutto quando si vuole andare ad esaminare e ad addestrare una macchina sfruttando maggiori quantità di dati e di conseguenza anche con fold più numerosi. Invece l'approccio full, nonostante più dispendioso in termini di risorse computazionali e di calcolo, risulta essere sempre più stabile ed efficiente in termini di accurattezza, sia lavorando su quantità di dati più elevati che su quantità di dati inferiore. Sul nostro test di 9 caratteri i risultati ottenuti sono:

1) approccio mean(con normalizzazione, C=0.1) accuracy = 100%
2) approccio mea(C=1) accuracy = 100%
3) approccio full (con normalizzazione C=0.001) accuracy = 100%
4) approccio full (C=0.001) accuracy = 100%

Ciò detto, per effettuare il test consigliamo il modello "full" (normalizzato o non con C=0.001) di cui abbiamo maggiore certezza, senza però avere alcun dubbio che anche con l'approccio "mean" si sarebbero ottenuti risultati in termini di accurattezza buoni.

Riferimenti bibliografici

[1] https://www.sciencedirect.com/science/article/pii/S1877065717304104
[2] https://ieeexplore.ieee.org/document/5642304