Machine Learning - AA 2020/2021

Report

Giuseppe Lasco
Dipartimento di Ingegneria dell'Informazione
Università degli studi di Roma "Tor Vergata"
Roma, Italia
giuseppe.lasco17@gmail.com

Marco Marcucci
Dipartimento di Ingegneria dell'Informazione
Università degli studi di Roma "Tor Vergata"
Roma, Italia
marco.marcucci96@gmail.com

I. Introduzione

Il progetto consiste nell'ottenere un modello di machine learning che, attraverso un'adeguata configurazione, permetta di ottenere il miglior risultato, rispetto alla metrica Accuracy, relativo ad un problema di classificazione multi-classe a 8 classi. Il compito da eseguire è relativo ad un problema di Hand Gesture Classification usando come input i dati storici generati dall'accelerometro a tre assi di un Wii Remote. I datasets forniti per addestrare il modello consistono in 5000 serie temporali, ognuna con 315 osservazioni nello spazio (x,y,z).

II. Progettazione

Per il raggiungimento dell'obiettivo, i dati fornitici vengono divisi in una parte di training costituita dall'80% delle istanze e una parte di testing contenente il restante 20%. La fase di progettazione si compone di due macro-fasi:

- Pre-processamento;
- Tuning degli iper-parametri.

Pre-Processamento

Nel machine learning la pre-elaborazione (o pre-processing) è la fase in cui i dati vengono preparati ed analizzati, prima di avviare l'algoritmo di apprendimento.

Questa fase si snoda in ulteriori step :

- Caricamento dati;
- Gestione valori nulli;
- Scaling dei dati;
- One-hot encoding.

Caricamento dati

Il caricamento dei dati in formato *csv* è stato effettuato attraverso la funzione `read_csv` della libreria `pandas.io.parsers`. Per creare il training set inseriamo i tre dataset (train_gesture_x.csv, train_gesture_y.csv, train_gesture_z.csv) all'interno di una lista e, attraverso la funzione `dstack` della librearia `numpy.lib.shape_base`, organizziamo i dati, all'interno di un tensore 3D, in modo tale che ogni singolo elemento consista in una terna di coordinate costituenti le 315 osservazioni di ognuna delle 5000 serie temporali.

Gestione valori nulli

Per evitare problemi in fase di apprendimento sono stati rimossi i valori nulli all'interno dei dataset. L'approccio utilizzato è stato quello di sostituire tali valori con l'occorrenza più vicina, in termini di osservazione, per ogni serie temporale, utillizzando le funzioni `fillna(method="ffill")` e `fillna(method="bfill")`. Nel caso in cui una determinata serie temporale contenga unicamente valori nulli, si procede all'eliminazione di quest'ultima per le tre coordinate e per la label. Nei dataset utilizzati come training non sono stati rilevati valori nulli.

Scaling dei dati

Lo scaling è una tecnica che permette di rendere i dati omogenei e indipendenti dall'unità di misura, in modo da individuare con più facilità le eventuali correlazioni tra i dati a disposizione.

In un primo momento la scelta è ricaduta su standardScaler e MinMaxScaler che sono comunemente più utilizzati. In seguito, per ampliare lo spettro di configurazioni possibili, si è scelto di inserire ulteriori tecniche di scaling quali RobustScaler e MaxAbsScaler. il primo sottrae al dato la mediana e lo scala utilizzando l'Inter Quartile Range, cioè l'intervallo di valori compresi tra il primo e il terzo quantile, mentre il secondo trasforma i valori, per ogni feature, in un intervallo compreso tra -1 e 1.

Lo scaler utilizzato nel modello finale è stato il RobustScaler.

One Hot encoding

Per fare in modo che la rappresentazione delle label, di tipo categoriche, risulti più espressiva è stata utilizzata una codifica one hot. Quest'ultima permette di rappresentare variabili categoriche come vettori binari. Quindi, ogni valore intero è rappresentato come un vettore binario che ha tutti valori a zero

tranne l'indice dell'intero considerato, che è contrassegnato con un 1.

Tuning degli Iper-parametri

L'obiettivo di questa fase è quello di ottenere la configurazione migliore di un particolare modello variando le possibili combinazioni degli iper-parametri. La funzione utilizzata per il tuning degli iper-parametri è `Hyperband` della libreria `keras_tuner`. Essa si basa su diversi parametri, tra cui i principali sono:

- **hypermodel**: rappresenta il modello da utilizzare;
- **objective**: rappresenta la metrica da massimizzare;
- **max_epochs**: rappresenta il numero di epoche massime;
- **factor**: rappresenta il fattore di riduzione per il numero di epoche e il numero di modelli;

Scelto il modello e una griglia di iper-parametri validi, questa funzione addestra quest'ultimo considerando un insieme di combinazioni degli iper-parametri e ricava la configurazione ottimale, in cross-validation, sulla base della metrica scelta. Sono stati utilizzati numerosi configurazioni di modelli per ottenere il miglior score possibile.

III. Modelli

Sono stati addestrati diversi modelli al fine di ottenere una accuracy soddisfacente:

- Convolutional Neural Network;
- Fully Connected Network;
- Multi-channel Deep Convolutional Neural Network;
- Inception Network.

Per ognuno di questi modelli, è stato effettuato tuning degli iper-parametri per identificare la migliore configurazione possibile. Tra questi, i risultati migliori sono stati ottenuti dal modello Convolutional Neural Network con la configurazione riportata in Tabella I.

TABLE I: Configurazione Convolutional Neural Network

Layer	Parametri
Conv1D	filter=64, kernel_size=10, activation="relu", l2_reg=0.01
MaxPooling1D	pool_size=8, strides=5, padding="same"
Conv1D	filter=64, kernel_size=8, activation="relu"
MaxPooling1D	pool_size=8, strides=5, padding="same"
Conv1D	filter=64, kernel_size=8, activation="relu"
MaxPooling1D	pool_size=4
Flatten	None
Dense	units=400, activation='relu'
Dropout	rate=0.2
Dense	units=100, activation='relu'
Dropout	rate=0.2
Dense	units=8, activation='softmax'

Le valutazioni sono state effettuate attraverso l'utilizzo di *k-fold-cross validation* con numero di fold pari a 8 e con un test set pari al 20% random del dataset originale. Al fine di evitare overfitting è stato utilizzato la tecnica di Early-Stopping con patiance pari a 30 epoche e con metrica di monitoraggio LOSS sul Validation Set. In Tabella II sono riportate i valori dell'accuracy del test set.

TABLE II: Accuracy Test Set

Modello	Min	Max	Average
Inception Network	0.961	0.979	0.974
Fully Convulutional Network	0.855	0.895	0.882
Convolutional Neural Network	0.975	0.985	0.980
Multi-Channel Deep Convolutional Neural Network	0.944	0.961	0.954

Valutazione sul Test Set

Il modello addestrato è stato salvato in un file di tipo `.h5`, mentre lo scaler è stato salvato in un file di tipo `.npy`. Nella fase di valutazione il modello e lo scaler vengono caricati e applicati sul test set.

IV. Osservazioni

Un'analisi preliminare del dataset ci ha permesso di osservare come quest'ultimo risulti essere bilanciato in termini di numerosità delle label, come è possibile osservare dalla Tabella III. Inoltre, queste ultime risultano essere equamente distribuite.

TABLE III: Numero label nel dataset

0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
631	635	613	600	638	645	597	641

V. Manuale d'uso

Per la descrizione della struttura del codice e il manuale d'uso riferirsi al README.pdf

References

[1] https://scikit-learn.org/stable/
[2] https://pandas.pydata.org/
[3] https://numpy.org/
[4] https://keras.io/api/
[5] https://www.tensorflow.org/api_docs/python/tf
[6] https://matplotlib.org/stable/contents.html
[7] https://github.com/hfawaz/InceptionTime
[8] https://arxiv.org/pdf/1809.04356.pdf
[9] https://stackoverflow.com/