Machine Learning A.A. 2019/2020: Relazione Progetto NILM

Autori:

Marco Balletti
Francesco Marino

22 luglio 2020

Indice

1 Introduzione

La presente relazione tratta dei modelli basati su reti neurali proposti per la risoluzione del *Non-Intrusive Load Monitoring (NILM)*. Lo scopo è quello di realizzare un'architettura di reti neurali in grado di predire, a partire da un valore (espresso in Watt) di consumo energetico totale di un'abitazione, il consumo relativo delle diverse apparecchiature, in particolare, in questo caso, di un frigorifero e di una lavastoviglie. Per la realizzazione del progetto è stato utilizzato *Python*, come linguaggio di programmazione, affiancato alle librerie *Numpy* (per la gestione efficiente di *array* di dati), *Pandas* (per il reperimento dei dati), *Tensorflow* e *Keras* (per la realizzazione e l'addestramento dell'architettura). La piattaforma di sviluppo utilizzata è *Google Colab* che mette a disposizione degli utenti la possibilità di addestrare delle reti neurali utilizzando anche delle *GPU* remote.

2 I dati

Per la realizzazione del progetto sono stati forniti tre *dataset*: il primo contiene i consumi dell'intera abitazione presa in esame registrati con una granularità pari ad un secondo, il secondo ed il terzo contenenti, invece, i consumi relativi al frigorifero e alla lavastoviglie registrati con la stessa granularità e nello stesso intervallo temporale del primo *dataset*. Visto l'ampio quantitativo di dati e la loro dimensione, si è ritenuto conveniente eseguirne il caricamento su *GitHub* in formato compresso (`.zip`), tale scelta è motivata anche dalla semplicità con cui, utilizzando *Python*, *Keras* e *Pandas*, è possibile trasformare un *dataset* compresso in un *dataframe*.

2.1 *Training*, *Validation* e *Test set*

Per garantire la correttezza dell'addestramento del modello e poterne valutare le prestazioni, è stato realizzato lo *split* del *dataframe* ricavato precedentemente in *training set*, *validation set* e *test set*. Il primo è stato utilizzato per eseguire l'addestramento dei modelli, il secondo per monitorarne l'*overfitting* e l'ultimo per valutare le metriche richieste (l'*energy precision*, l'*energy recall* e l'*energy F1*) sul modello e poter confrontare le differenti possibili soluzioni tra loro. Durante la creazione di questi *set* e la conversione da *dataframe* ad *array Numpy*, si pone particolare attenzione al consumo della memoria RAM, vengono, infatti, eliminati tutti i riferimenti alle strutture dati non più necessarie dopo la conversione in modo da permettere al *garbage collector* di eliminarle e recuperare memoria.

Poiché i dati in questione sono di tipo serie temporali, si è deciso di realizzare il *validation set* ed il *test set* con misurazioni che fossero temporalmente successive a quelle presenti all'interno del *training set*, tale relazione di ordine si mantiene anche tra *validation* e *test set*.

2.2 Normalizzazione

I dati così suddivisi vengono, quindi, sottoposti a normalizzazione sottraendo loro il valore medio e dividendoli per la deviazione standard, tale procedimento si è rivelato essere una strategia particolarmente efficace per la riduzione dei tempi di addestramento dei modelli. I valori necessari per effettuare questa trasformazione sui dati sono stati calcolati utilizzando il solo *training set*: si è proceduto, in particolare, calcolando la media e la deviazione standard dei consumi dell'abitazione, del frigorifero e della lavastoviglie su questo *set* di dati, il *training set* ed il *validation set* normalizzati sono stati ottenuti, quindi, andando a sottrarre la media relativa al tipo di dato (totale, frigorifero o lavastoviglie) e dividendo per la corrispondente deviazione standard precedentemente calcolate. Per quanto concerne il *test set*, l'operazione di normalizzazione è stata svolta solo sui consumi totali (sempre utilizzando i valori ricavati dal *training set*), non è stato necessario eseguirla sui corrispondenti valori di frigorifero e lavastoviglie poiché i dati ottenuti dalla predizione sul *set* in questione vengono denormalizzati (moltiplicando per la deviazione standard e sommando la media del *training set*) prima di eseguire la valutazione delle prestazioni.

2.3 I generatori

L'addestramento e la valutazione delle prestazioni di modelli basati su dati di tipo serie temporali richiede l'utilizzo di finestre, queste strutture di dati risultano essere necessarie sopratutto se ci si trova nei casi di predittori *sequence to sequence* o *sequence to point*. Realizzare una finestra temporale nel modo classico (`for` *loop*) si rivela essere particolarmente sconveniente sia in termini di prestazioni che in termini di consumo di memoria RAM, quest'ultimo aspetto, in particolare, risulta essere ovvio se si considerano finestre temporali parzialmente sovrapposte per cui una stessa misurazione che ricade in più finestre temporali viene memorizzata molteplici volte.

Per ovviare a questa problematica ed evitare di incorrere in *crash* dell'*environment Google Colab*