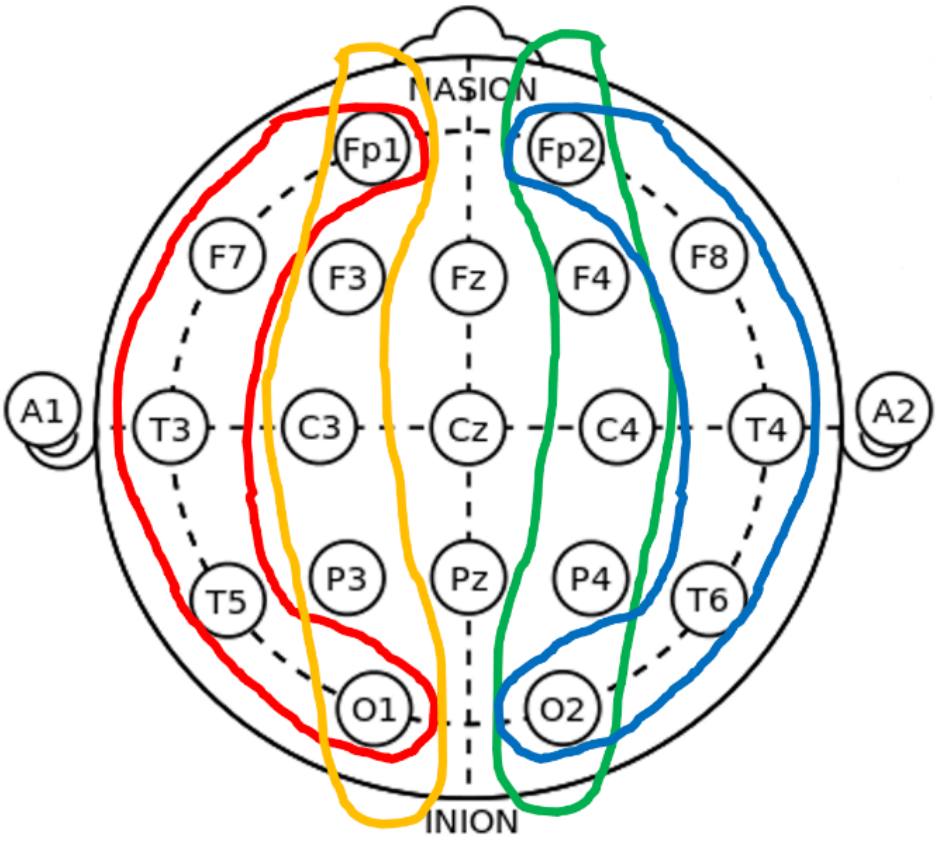
AutoML for Brain Predictor

UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

Classificazione dei pattern di attività cerebrali dannose in pazienti critici.



- **LL - Left Temporal Chain**
- **LP - Left Parasagittal Chain**
- **RP - Right Parasagittal Chain**
- **RR - Right Temporal Chain**

Alessandro Isceri - 879309

Mattia Ingrassia - 879204

Introduzione

Il progetto consiste nell'analizzare un dataset contenente segnali elettroencefalografici (EEG) e spettrogrammi utilizzando AutoML come tecnica di Machine Learning.

Nello specifico, il dataset utilizzato è stato preso da una competizione di Kaggle e il framework AutoML usato è AutoPyTorch.

Il dataset contiene le valutazioni fornite da alcuni esperti sulla base di diversi campioni EEG della durata di 50 secondi con i rispettivi spettrogrammi ricavati da pazienti ospedalieri critici.

L'obiettivo è stimare, attraverso l'uso della regressione, sei valori target:

- seizure_vote
- lpd_vote — lpd = lateralized periodic discharges
- gpd_vote — gpd = generalized periodic discharges
- lrda_vote — lrda = lateralized rhythmic delta activity
- grda_vote — grda = generalized rhythmic delta activity
- other_vote

Ognuno di questi numeri reali rappresenta la frequenza relativa dei voti ottenuti per ciascuna categoria, ed è dunque un numero compreso tra 0 ed 1.
Inoltre, la somma delle frequenze relative deve essere pari ad 1.

Descrizione del dataset

Il dataset è diviso in tre parti:

- train.csv, contiene i metadati che permettono di estrarre i sottoinsiemi originali che sono stati annotati dagli esperti.
- train_eegs, è una cartella contenente dei file .parquet, in particolare, è presente un file per ogni EEG contenente le varie misurazioni.
- train_spectrograms, è una cartella contenente dei file .parquet, in particolare, è presente un file per ogni spettrogramma contenente le varie misurazioni.

Sono stati utilizzati dei dataset forniti da un utente di Kaggle, Chris Deotte, per velocizzare la lettura dei dati presenti nelle due cartelle.

- https://www.kaggle.com/datasets/cdeotte/brain-eeg-spectrograms/data
- https://www.kaggle.com/datasets/cdeotte/brain-spectrograms

Sviluppo

L'idea iniziale era quella di organizzare il dataset in modo tale da utilizzare delle immagini per effettuare la regressione multipla sulle sei colonne target.

Tuttavia, AutoPyTorch non supporta la regressione sulle immagini, di conseguenza, è stato deciso di cambiare approccio.

Dato che AutoPyTorch supporta la regressione singola su dati tabulari, prendendo spunto da alcuni Notebook di altri utenti, i dati sono stati organizzati come un'unica grande tabella.

Visto che i dati, come spiegato nella sezione precedente, sono divisi in tre dataset, il primo step consisteva nel raggruppare le informazioni.

L'elaborazione dei dati viene svolta dalla funzione *load_data*, il cui codice è disponibile qui.

L'ambiente utilizzato è un notebook di Kaggle, che fornisce un ambiente di sviluppo con python preinstallato, 30 GB di RAM, diverse possibilità di scelta per le schede grafiche e sessioni e run dalla durata massima di 12 ore.

Data la notevole mole di dati, l'utilizzo della RAM era particolarmente elevato, per permettere ad AutoPyTorch di lavorare senza problemi di memoria, è stato creato un sottoprocesso con lo scopo di elaborare i dati in modo tale che, una volta terminato, la RAM venisse liberata da tutte le variabili non più necessarie. In questo modo, l'utilizzo della RAM è sceso dal 65% al 7%.

Il codice riguardante la gestione dei processi è disponibile qui.

Una volta organizzati correttamente i dati a disposizione, è stato possibile proseguire con la costruzione dei modelli.

CatBoost

Prima di utilizzare AutoPyTorch, il formato dei dati è stato testato con un modello "classico" di Machine Learning per capire se fosse adatto al task.

Il modello è stato realizzato usando il framework CatBoost (codice), che ha portato ai seguenti risultati:

```
CatBoost Starter - By Alessandro Isceri & Mattia Ingrassia
R^2  seizure_vote  :  0.40043432542414814
R^2  lpd_vote  :  0.551633882327218
R^2  gpd_vote  :  0.6539692953443332
R^2  lrda_vote  :  0.25797722817477164
R^2  grda_vote  :  0.387760890281311
R^2  other_vote  :  0.4556661141739473
```

Per fare un confronto, riportiamo i risultati di un notebook con LB score pari a 0.43:

```
EfficientNetB2 Starter - By Chris Deotte - LB 0.43
R^2  seizure_vote  :  0.5558249098570601
R^2  lpd_vote  :  0.5797308538791207
R^2  gpd_vote  :  0.6562397283510863
R^2  lrda_vote  :  0.2295797875912058
R^2  grda_vote  :  0.41246918413123246
R^2  other_vote  :  0.5265252726434566
```

Data la complessità del dataset e i risultati ottenuti da altri partecipanti, si possono considerare accettabili i risultati ottenuti con questo starter CatBoost.

AutoPyTorch

Dopo aver confermato il formato dei dati grazie a CatBoost, si è passati all'utilizzo di AutoPyTorch.

Le ultime modifiche del framework risalgono al 2022, di conseguenza, ci sono stati dei problemi durante l'installazione causati dalle dipendenze con le versioni di alcune librerie.

Per risolvere il problema, è stato utilizzato un branch creato da Borda che modifica i requisiti di installazione cambiando le dipendenze per le librerie interessate.

```
!pip install "git+https://github.com/Borda/Auto-PyTorch.git@bump/sklearn-1.0+"
```

Dopo aver caricato i dati analogamente a come fatto con CatBoost, essi sono stati divisi in 4 insiemi: X_train, y_train, X_test, y_test utilizzando la funzione *train_test_split* offerta da sklearn (codice).

In un secondo momento, dato che AutoPyTorch non supporta la regressione multipla, è stata effettuata una ricerca del modello migliore per ogni singola colonna target (codice).

La ricerca del modello è organizzata in due fasi:

- Nella prima fase, AutoPyTorch cerca gli algoritmi che performano meglio sul dataset, e crea un ensemble usando gli algoritmi trovati (codice).
- Nella seconda fase, AutoPyTorch effettua un refit usando l'intero dataset sull'ensemble creato e stampa a video la configurazione dell'ensemble (codice).

AutoPyTorch non supporta l'utilizzo delle GPU come discusso in questa sezione nella repository ufficiale su GitHub, quindi le run sono state interamente eseguite sulle cpu.

Di conseguenza, dato che il dataset è abbastanza complesso, le run hanno impiegato molte ore portando però a buoni risultati in confronto a quelli ottenuti con CatBoost.

Risultati

Di seguito vengono riportati i risultati ottenuti e gli ensemble trovati:

SEIZURE_VOTE

{'r2': 0.551919909865034}

	Preprocessing	Estimator	Weight
0	None	CBLearner	0.56
1	None	LGBMLearner	0.44

LPD_VOTE:

{'r2': 0.6659664390967204}

	Preprocessing	Estimator	Weight
0	None	CBLearner	0.56
1	None	LGBMLearner	0.38
2	None	KNNLearner	0.06

GPD_VOTE:

{'r2': 0.7318632825802545}

	Preprocessing	Estimator	Weight
0	None	CBLearner	0.72
1	None	LGBMLearner	0.2
2	None	KNNLearner	0.08

LRDA_VOTE:

{'r2': 0.34247518725730186}

	Preprocessing	Estimator	Weight
0	None	CBLearner	0.44
1	None	LGBMLearner	0.34
2	None	KNNLearner	0.22

GRDA_VOTE:

{'r2': 0.4827942239451889}

	Preprocessing	Estimator	Weight
0	None	CBLearner	0.46
1	None	LGBMLearner	0.44
2	None	KNNLearner	0.1

OTHER_VOTE:

{'r2': 0.5174849184721397}

	Preprocessing	Estimator	Weight
0	None	CBLearner	1

Per migliorare i risultati ottenuti, si è pensato di utilizzare la strategia k-fold, tuttavia, a causa dei problemi citati in precedenza, il tempo di run era troppo elevato e quindi non c'è stato alcun miglioramento.

Conclusioni e possibili sviluppi futuri

La disponibilità delle GPU avrebbe sicuramente velocizzato AutoPyTorch nella ricerca del modello ideale e nella scelta degli iperparametri; infatti, confrontando le tempistiche di CatBoost con e senza schede video, una run passava da 10 ore utilizzando solo la CPU a 20 minuti con il supporto di 2 schede video Nvidia T4.

Inoltre, l'aumento della potenza computazionale avrebbe permesso l'utilizzo della strategia k-fold per provare a migliorare ulteriormente i risultati.

La mancanza di aggiornamenti nell'ultimo periodo limitano fortemente l'utilizzo di questo framework, ad esempio, si potrebbe implementare la regressione multipla o la regressione sulle immagini e il supporto per le GPU.

AutoPyTorch è un ottimo strumento, che permette di ottenere buoni risultati anche a chi non ha quasi alcuna conoscenza di Machine Learning, a discapito degli sviluppatori più esperti che sono limitati alle opzioni di configurazioni offerte dal framework per la scelta o iper-parametrizzazione degli algoritmi.