Classificazione tramite I.A. del Sapore di Jet Adronici

Progetto Finale n.6 del corso di Metodi di Intelligenza Artificiale e Machine Learning *(Canale M-Z), A.A. 2021/2022*

L. Pietropaoli, A. Scarpa

Sommario

Scopo di questo progetto è di mostrare come sia possibile implementare, tramite algoritmi di machine learning relativamente *semplici*, classificatori del *quark flavor* di jet di particelle prodotti nelle collisioni adroniche - con prestazioni notevoli. In questa trattazione ci si interessa in particolare a discernere i jet formati da particelle a sapore *leggero* (quark u, d, s o gluoni) - che saranno considerati gli eventi di segnale - dai jet originati da altre particelle (in particolare da quark c o b. Questi ultimi sono quark a sapore *pesante*), che la classificazione dovrà riconoscere come segnale di rumore. Il modo in cui si procederà è il seguente: dopo una più generalizzata trattazione del campione di dati preso in esame, si implementeranno due modelli per la classificazione di eventi segnale e *noise*, in particolare una rete neurale feedforward con layer densi e una rete neurale "Long Short Term Memory" o LSTM.

Il dataset usato, i notebook prodotti per questa trattazione e le referenze citate sono reperibili qui.

1 Analisi del dataset

Il dataset di cui si dispone per l'esperienza consiste in una simulazione di produzione di jet adronici per un totale di circa 11 milioni di jet, ossia 11 milioni di righe.

Ogni riga contiene le misurazioni di un jet sotto forma di quattro categorie principali di variabili:

- **jet $\mathbf{p_T}$**: momento trasverso del jet rispetto al fascio di particelle
- **jet eta**: pseudorapidità, ovvero l'angolo fra la particella prodotta e il fascio
- **variabili di traccia**: un vettore di dimensione 8 contentente le misurazioni di variabili relative alla traccia del jet
- **variabili di vertice**: un vettore di dimensione 6 contenente le misurazioni di variabili relative ai *jet vertices*; queste variabili sono ricostruite a partire dalle tracce [1]

Queste quattro variabili (o classi di variabili, come per le *track variables* e le *vertex variables*) rappresentano per il problema studiato **feature di alto livello**.
Essenziale per la classificazione è avere nel dataset una colonna di **label**, individuata nel *flavor* del jet (light $\longrightarrow$ 0, charm $\longrightarrow$ 4, heavy $\longrightarrow$ 5), che corrisponde alla terza colonna del dataset, posizionato dopo jet p_T e jet eta.

Altre variabili di traccia *e* vertice (di **basso livello**) sono state scartate dal dataset poiché - a fronte di un riscontro con altri studi [1] condotti sullo stesso dataset - sono state ritenute appesantire l'allenamento e la classificazione dei modelli in maniera ingiustificata rispetto all'incremento nelle prestazioni che vi si correlava.

[1] Questa è un'informazione importante poiché suggerisce che tra feature di traccia e feature di vertice saranno presenti correlazioni.

1.1 Lettura

Il primo step è stato la lettura del dataset: si è scelto di lavorare, piuttosto che sull'intero campione, sulle prime 800 mila righe di dati [2]. Sul dataset sono state apportate alcune modifiche per renderlo più "maneggiabile" a livello computazionale:

1. Poiché le feature relative alle tracce sono contenute in una matrice di dimensione (800k, 8), essa è stata trasfromata in 8 vettori di dimensione 800k.
2. Poiché le feature relative ai vertici sono contenute in una matrice di dimensione (800k, 6), essa è stata trasfromata in 6 vettori di dimensione 800k.

In questo modo si è ottenuto un *dataframe* contenente 800k eventi disposti in 16 colonne di feature di alto livello e una colonna di label.

1.2 Cleaning

La seconda fase di preparazione del campione di dati è consistita nell'individuare **valori anomali** e trasformarli in modo da **renderli interpretabili e riconoscibili** da un algoritmo di machine learning.

Alcune feature continue contengono valori `inf` e `-inf`: tali valori sono presenti nel dataset originario per indicare che per quella specifica collisione il valore della variabile può essere mancante, non definito o non misurabile fisicamente. *Per esempio, nei jet con light flavor spesso non è possibile ricostruire il vertice a partire dalla traccia.*

Ancora: alcune variabili discrete e/o definite positive assumono valore `-1`, anch'esse a indicare dati mancanti, non definiti o non misurabili [3].

[2] Il *notebook environment* - Google Colab - in cui è stato svolto tutto il lavoro riguardante questo progetto limita la capacità della RAM utilizzabile.

[3] Tutta la documentazione che giustifica queste considerazioni è disponibile nel link in calce al Sommario.

Infine nel dataset considerato è stato individuato anche un valore `NaN`, solitario: stessa origine dei valori `inf` e `-inf` e `-1`.

I valori `inf` e `-inf` sono stati sostituiti con `-1` in quanto è molto problematico lavorare con gli infiniti, ma soprattutto per accorpare tali punti alla seconda classe di valori non fisici. Così facendo saranno visibili al classificatore, che avrà modo di riconoscere l'anomalia (letteralmente, la diversità) di questi dati.

È stato scelto il valore `-1` per accorpare tutti i dati in corrispondenza di un punto isolato rispetto alle distribuzioni di tutte le altre variabili. Inoltre ciò era conveniente perché non c'è stato bisogno di sostituire i valori che erano già `-1`.

La percentuale di valori non fisici per alcune variabili superava anche il 30%, come per le variabili `delta r vertex` *e* `vertex energy fraction`.

1.3 Statisica

Le **distribuzioni** dei valori assunti dalle feature possono essere visualizzate con degli istogrammi (Figura 4) - in cui a colori diversi corrispondono diversi jet flavor - e forniscono un primo strumento di analisi del campione:

- ✎ si nota come sia **maggiore l'abbondanza di eventi di segnale** (in rosso) rispetto al fondo di quark leggeri e c-quark (si veda ad esempio la distribuzione del `jet eta`);
- ✎ che sia correlata o meno all'abbondanza relativa degli eventi nelle tre classi, si nota come per diverse variabili gli **eventi di segnale** siano distribuiti su un **range di valori più ampio** rispetto a quelli di fondo. Questo non vale, chiaramente, per le variabili di angolo (come ad esempio `jet eta`) o di probabilità, ma si può osservare bene nel **numero di tracce di vertici secondari** o nella **massa dei vertici**.

Inoltre per visualizzare le **correlazioni** fra le colonne del dataset, ossia tra le variabili misurate, è stata graficata la *scatter matrix* (visibile in Figura 1) fra di esse.

In particolare, si sono selezionate per la visualizzazione le feature più correlate tra loro (che formano una matrice di 8×8 plot): tali feature sono state scelte in una prima analisi su un campione di 100 collisioni, scartando quelle che apparivano meno correlate.

Delle rimanenti è stata plottata la matrice di correlazione per 10k eventi, da cui si osserva che:

- ✎ le variabili di **significanza di traccia** (sono 4), e in particolare le prime due (nell'ordine in cui compaiono nel dataset e quindi nella matrice), presentano una forte correlazione tra loro;
- ✎ le 4 variabili di significanza di traccia appaiono correlate con le seguenti variabili: `jet pt`, `vertex significance`, `vertex mass` e `vertex energy fraction` - anche qui, lo sono maggiormente `track 2 d0 significance` e `track 3 d0 significance`;
- ✎ c'è una fortissima correlazione tra **massa** e **frazione di energia** nei vertici, che è fisicamente aspettato.

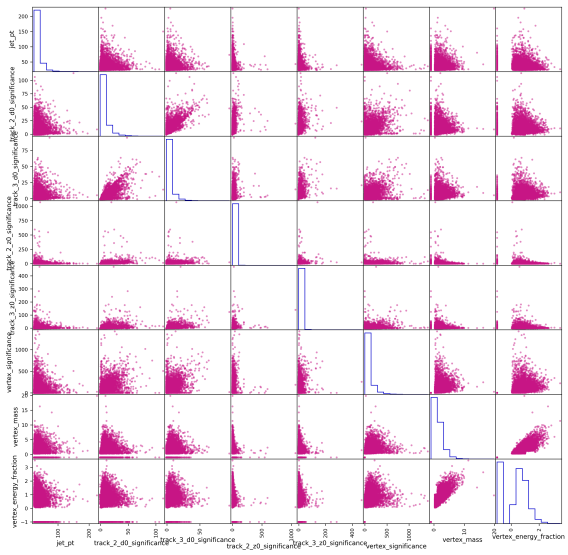

Figura 1: Matrice di correlazione per le variabili più correlate.

2 Classificazione

Venendo all'obiettivo principale di questo studio, si sono addestrati due modelli di machine learning con la task di classificare gli eventi come segnale o rumore, avendo a disposizione l'intera matrice delle feature e il vettore di label come *target*. In particolare riguardo il vettore di target, in linea con l'obiettivo di effettuare una classificazione binaria, si è schiacciata la tridimensionalità della variabile `flavor` su un vettore sostitutivo con due possibili valori: `1` se segnale, `0` altrimenti.

Il campione di dati è stato suddiviso nel seguente modo: l'80% (640k eventi) per il training, il 10% (80k eventi) per la validazione e il restante 10% (80k eventi) per il testing.

Successivamente i valori delle feature sono stati normalizzati, cioè distribuiti intorno a 0 con deviazione standard 1. *I valori di medie e deviazioni standard delle singole feature utilizzate per la normalizzazione sono stati calcolati sul campione di training.* [4]

2.1 Shallow MLP (Feedforward Neural Network)

Il primo modello addestrato è stata una rete shallow MLP (Multi Layer Perceptron) con layer densi. La rete implementata è costituita da 9 layer densi feedforward con uscita sigmoide: il primo layer ha 16 neuroni in input (dev'essere chiaramente pari al numero di feature), i layer nascosti hanno $N = 400$ [5] neuroni l'uno, l'ultimo dei quali ha 2 neuroni in output (per la classificazione segnale-background); inoltre è stato applicato un dropout con probabilità $p = 0.3$ [6] ai primi due layer.

Il training è consistito in un loop su 100 epoche per l'ottimizzazione degli iperparametri - parametri interni della rete

[4] Per evitare information leaks nei campioni destinati alla valutazione delle prestazioni dei modelli (validation e test).

[5] Questo valore è stato scelto in seguito a un'ottimizzazione manuale.

[6] Scelto nello stesso modo in cui è stato scelto il numero di neuroni degli *hidden layer*.

neurale - attraverso una SDG (*Stochastic Gradient Descent*) con learning rate LR = 0.01 (un valore di inizializzazione piuttosto standard) e *momentum* $m = 0.9$. [7] Nelle ultime 20 epoche il LR è stato fatto decrescere fino ad un valore di 0.001.

Per quanto riguarda la valutazione delle prestazioni del modello in fase di training, è stata scelta come loss function la Cross Entropy Loss e come metrica l'*accuracy*. A tal proposito, si riportano in Figura 2 gli andamenti della training/validation loss e della training/validation metric in funzione del numero di epoche. Gli iperparametri scelti dal modello stesso sono stati quelli che minimizzano la loss nel campione di validation: non è infatti immediato, per un MLP su 100 epoche (relativamente poche), che al crescere del numero di epoche la loss si minimizzi senza fluttuazioni importanti. Per questo è stata implementata una funzione che salvasse il miglior modello se a una determinata epoca la loss fosse minore di quella calcolata all'epoca precedente. *Si vedrà in sottosezione 2.2 che per una LSTM già con 50 epoche si ottiene una minimizzazione della loss in funzione del numero di epoche molto meno oscillante.* Con il miglior modello si è poi fatta inferenza per testare le prestazioni della rete su un campione "mai visto prima", e per valutarne le prestazioni sono state utilizzate due metriche di test: l'AUC (ROC *Area Under Curve* [8]), e l'accuracy.

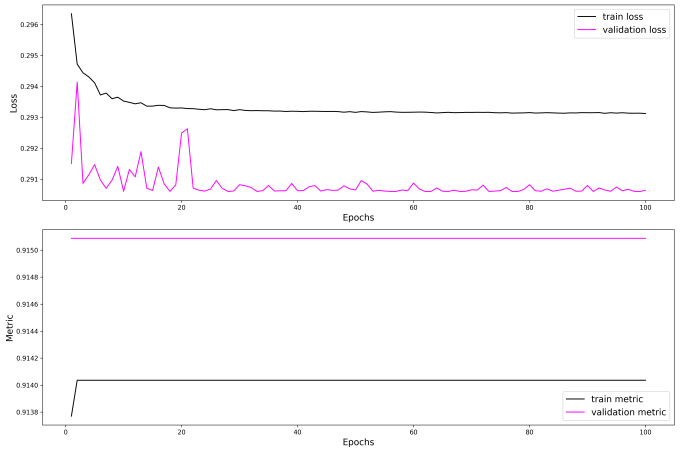

Figura 2: Prestazioni durante il training della rete neurale MLP. Si vede come la fluttuazione della validation loss sia importante per le prime ≈ 20 epoche, per poi tendere a stabilizzarsi.

2.2 Rete LSTM

Il secondo modello implementato è una rete LSTM (Long Short Term Memory) a layer singolo. La rete presenta un singolo layer composto da $N = 70$ neuroni e un layer feed-forward con 2 neuroni di output (sempre per la classificazione segnale-background). Il loop di training è stato svolto su 50 epoche e per l'ottimizzazione degli iperparametri è stata eseguita una discesa stocastica lungo il gradiente con learning rate LR = 0.01 (stavolta fisso nel tempo) e inerzia $m = 0.9$. Anche in questo caso la loss utilizzata è stata la Cross Entropy e la metrica per valutare la fase di allenamento l'accuracy. In Figura 3 sono graficati gli andamenti della training/validation loss e della training/validation metric in funzione del numero di epoche. Poiché il comportamento della loss è evidentemente più lineare durante la discesa lungo il gradiente per una rete LSTM con i parametri scelti rispetto all'MLP implementato in precedenza, gli iperparametri che minimizzano la loss sono proprio quelli prodotti all'ultima epoca. Per la valutazione delle prestazioni in fase di test si sono impiegate nuovamente l'AUC e l'accuracy.

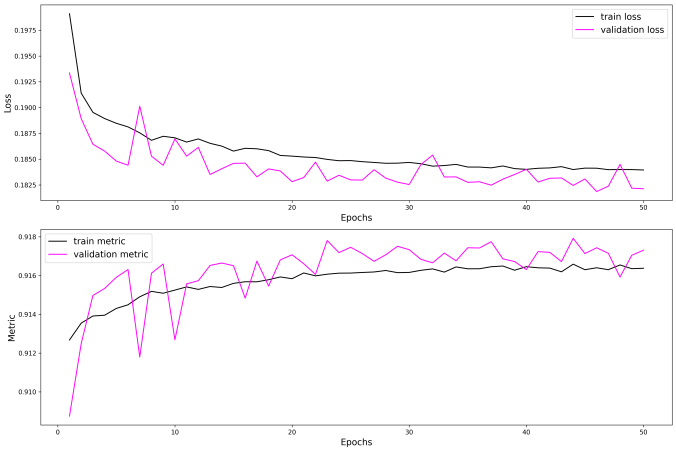

Figura 3: Prestazioni durante il training della rete neurale LSTM. Osserviamo che la fluttuazione della validation loss, sia piuttosto uniforme nel tempo, con una dipendenza molto meno forte dal numero di epoche rispetto al caso di rete densa shallow. Questa caratteristica ci permette di prendere il miglior modello come la sua versione all'ultima epoca.

3 Risultati e Conclusioni

Si riportano in Tabella 1 i risultati ottenuti in termini di metrica in fase di testing per entrambi i modelli:

- facendo un confronto con l'articolo di riferimento [1], il valore di AUC per **MLP** risulta essere di circa il 6% **peggiore**. [9] Ciò potrebbe essere dovuto a una non perfetta ottimizzazione degli iperparametri;
- il valore di AUC per **LSTM** invece risulta **compatibile** [10] con il valore di riferimento;
- esaminando infine l'**accuracy**, si vede come entrambi i modelli siano prestanti a ben **oltre il 90%**.

Poiché l'obiettivo era di dimostrare come in modo *semplice* si possono implementare algoritmi di machine learning che funzionino come classificatori (le reti neurali svolgono questo compito su grandi quantità di dati in modo molto efficiente), possiamo ritenerci soddisfatti dei risultati ottenuti, in quanto entrambe le reti implementate forniscono prestazioni notevoli.

	MLP	**LSTM**
AUC	0.853	0.915
accuracy	0.914	0.917

Tabella 1: Valori delle metriche (AUC (Area Under ROC curve) e accuracy) ottenuti per i due modelli neurali implementati.

[7] Un parametro d'inerzia che favorisce la direzione di discesa lungo il gradiente ottimizzata allo step precedente.

[8] Cioè l'area sotto la *Receiver Operating Characteristic curve* (ROC).

[9] Il confronto è effettuato a parità di feature impiegate.

[10] Tenendo comunque presente che non viene fornita una stima dell'incertezza su tale risultato.

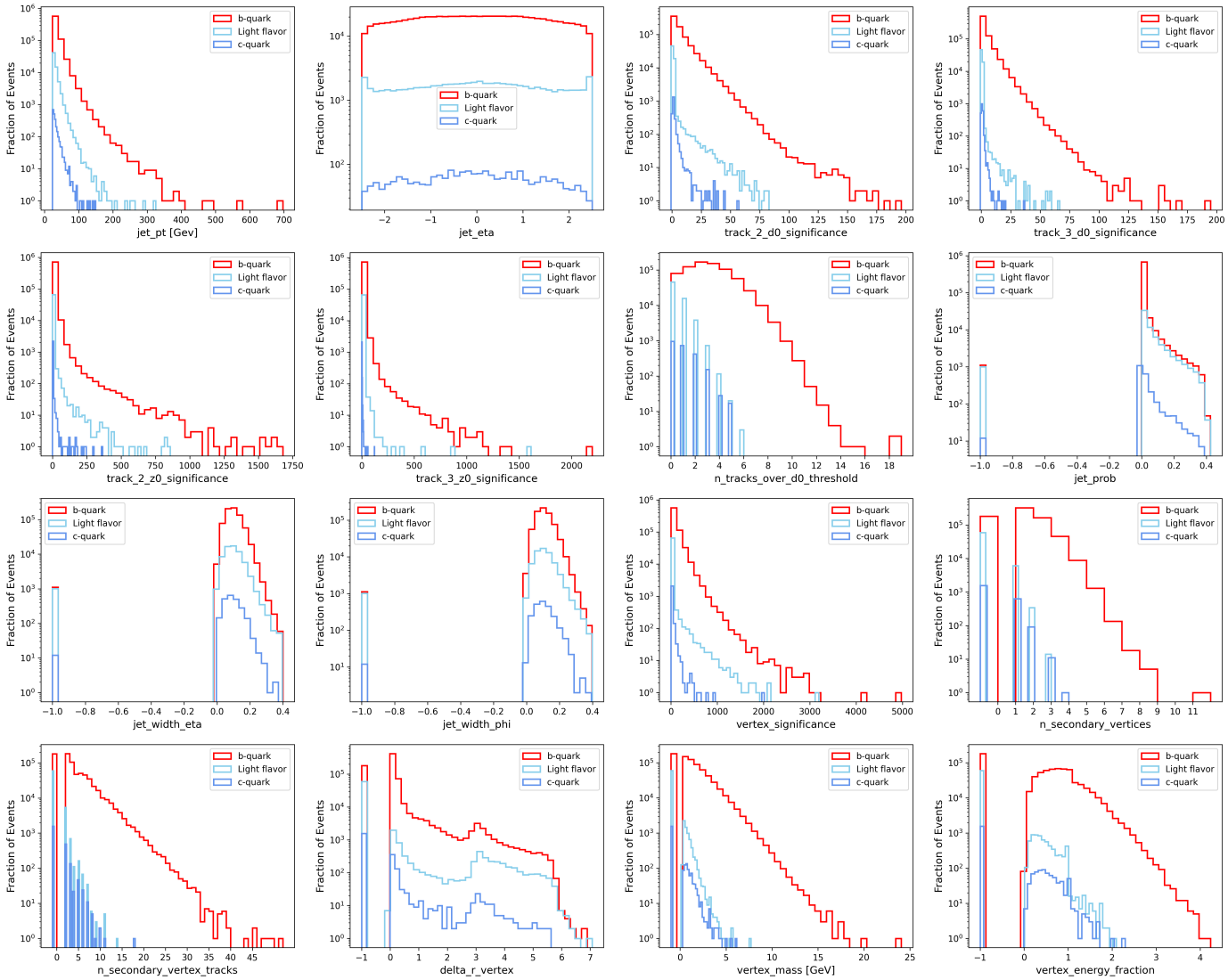

Figura 4: Istogrammi delle distribuzioni delle **16 feature di alto livello** sul campione esaminato di 800k collisioni adroniche. Per ogni feature, in rosso sono indicati gli eventi di segnale (heavy flavor, ossia quark bottom), in azzurro e in blu gli eventi di background (rispettivamente light flavor e quark charm). Si nota un'importante percentuale di eventi anomali, tutti distribuiti in corrispondenza del valore `-1` come voluto (si veda sottosezione 1.2).

Riferimenti bibliografici

[1] D. Guest, J. Collado, P. Baldi, S. C. Hsu, G. Urban, D. Whiteson. *Jet Flavor Classification in High-Energy Physics with Deep Neural Networks*, Phys. Rev. D 94, 112002 (2016). [link]