15 Dicembre 2022

Esercitazione n. 2
Metodi Statistici di Analisi Dati

Esercizio 1

Per gli esercizi seguenti, riportare nella soluzione le risposte numeriche e i grafici richiesti e solo quei pezzi di codice in cui hai apportato modifiche non banali; non inviare lunghe stampe di codice che sono essenzialmente uguali ai programmi originali.

Esercizio 1: per questo esercizio eseguirai una semplice analisi multivariata con il pacchetto TMVA

https://root.cern/manual/tmva/

insieme alle routine ROOT. Usando il codice a questo link:

https://cernbox.cern.ch/s/BZQRndBcTaTfQCK

Come al solito, per creare i programmi, digita `gmake`. Le librerie ROOT devono essere installate; se questo non funziona, per favore chiedi aiuto. Utilizza ROOT 6.

Innanzitutto, utilizzare il programma `generateData` per generare due n-tuple di dati con 10.000 eventi ciascuno i cui valori seguono una determinata distribuzione tridimensionale per l'ipotesi del segnale e un'altra per l'ipotesi di fondo. Le n-tuple vengono create e memorizzate usando la classe `TTree` i `ROOT`. Tre quantità (x, y, z) sono misurate per ogni evento. Usando la macro `makeScatterplots.C`, date un'occhiata ad alcune delle distribuzioni (eseguite root e digitate `.x makeScatterplots.C`).

Quindi utilizzare il programma `tmvaTrain` per determinare i coefficienti di un discriminante di Fisher. Quando si esegue il programma, i coefficienti delle funzioni discriminanti vengono scritti in una sottodirectory `dataset/weights` come file di testo. Dai un'occhiata a questi file e identifica i coefficienti pertinenti.

Infine usa il programma `analyzeData` per analizzare i dati generati. Supponiamo di voler selezionare eventi di segnale e che le probabilità a priori di segnale e fondo siano uguali.

i) Supponiamo di selezionare eventi segnale richiedendo $t_{\text{Fisher}} > 0$. Qual è l'efficienza del segnale (potenza del test) e del fondo (dimensione del test), cioè, $\varepsilon_s = P(t_{\text{Fisher}} > 0 \mid s)$ e $\varepsilon_b = P(\text{tFisher} > 0 \mid b)$? Qual è la purezza del segnale, cioè $P(s \mid t_{\text{Fisher}} > 0)$? (Inserisci il codice in `analyzeData.cc` per contare il numero di eventi di segnale e di fondo che sono selezionati).

ii) Crea istogrammi di t_{Fisher} per entrambi gli eventi di segnale e di fondo.

iii) Modificare i programmi `tmvaTrain.cc` e `analyzeData.cc` per includere un multilayer perceptron con un livello nascosto contenente 3 nodi. Per definire il multilayer perceptron è necessaria una riga del tipo:

```
factory->BookMethod(TMVA::Types::kMLP, "MLP", "H:! V: HiddenLayers = 3");
```

Vedere il manuale TMVA per maggiori dettagli. Questo memorizzerà i coefficienti del multilayer perceptron in un file nella sottodirectory `dataset/weights`.

Successivamente per analizzare i dati utilizzando il multilayer perceptron, sarà necessario aggiungere una chiamata a `reader->BookMVA` utilizzando i nomi corrispondenti (sostituire `Fisher` con `MLP`). Quindi definisci e riempi altri due istogrammi per la statistica MLP sia per

l’ipotesi di segnale che per l'ipotesi di fondo (fai questo in analogia con gli istogrammi per il discriminante Fisher). Crea grafici degli istogrammi risultanti utilizzando il campione di test.

Infine, seleziona gli eventi segnale richiedendo t_{MLP}> 0.5. Qual è l'efficienza del segnale e del fondo? Qual è la purezza del segnale assumendo pari probabilità a priori per i due tipi di eventi?

Esercizio 2

Questo esercizio è una continuazione di Es. 1.

2 (a) Modificare il programma `tmvaTrain.cc` per includere un Boosted Decision Tree (BDT) con una linea

```
factory-> BookMethod(TMVA::Types::kBDT, "BDT200", "NTrees = 200: BoostType = AdaBoost”);
```

Questo creerà un BDT con 200 iterazioni di boosting.

Successivamente, modificare il programma `analyzeData` per valutare il valore di BDT per ciascun evento e crearne degli istogrammi (mostrare le distribuzioni di segnale e di fondo sullo stesso grafico) sia per il campione di training che per il campione di test, statisticamente indipendente.

2 (b) Ora ripeti la procedura per diversi numeri di iterazioni di boosting, ad esempio 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 10,000, 50,000. A seconda del tuo livello di abilità con C ++ potresti desiderare di automatizzarlo in un ciclo. Nota che per fare questo, ogni classificatore deve avere un nome univoco di tipo stringa (nell'esempio sopra il nome è BDT200).

Per ogni classificatore, calcola il tasso di errore totale utilizzando un valore limite di $t_{cut} = 0$. Cioè, calcola la frazione di eventi (segnale e fondo) che si trovano sul lato sbagliato del limite. Riportare in un plot questa frazione in funzione del numero di iterazioni di boosting, sia per il campione di training che per il campione statisticamente indipendente di test. Determina approssimativamente il numero ottimale di iterazioni di boosting.