Advanced Machine Learning: Assignment #5

Fabrizio D'Intinosante — 838866

Università degli Studi di Milano Bicocca — December 9, 2019

Introduzione

L'obiettivo di questo *assignment* è quello di effettuare *hyper-parameters optimization* (HPO) di una rete neurale con lo scopo di massimare la sua *accuracy* in *10-fold cross validation*. Il task è composto da due precisi step:

- Partendo da una rete neurale composta da 2 *hidden layers* da 4 e 2 neuroni, ottimizzare il *learning rate* in un *range* [0.01, 0.1] ed il *momentum* in un *range* [0.1, 0.9]. L'ottimizzazione deve avvenire sfruttando due modelli scegliendo tra il *Gaussian Process* ed il *Random Forest* come modello surrogato e due funzioni di acquisizione distinte. Il numero di iterazioni previsto è 20, partendo per entrambi i modelli dalle medesime 5 configurazioni iniziali. Oltre ai due modelli di *Sequential Model Based Optimization* (SMBO), è richiesta anche la creazione di una rete ottimizzata sui medesimi parametri utilizzando 25 configurazioni di *Grid Search* e 25 iterazioni di *Random Search*;
- Il secondo step dell'*assignment* consiste nel ripetere quanto fatto al punto precedente ma ottimizzando anche il numero di neuroni dei due *layers* della rete neurale in un *range* [1, 5] definendo sempre due modelli con due funzioni di acquisizione distinte ma fissando come modello surrogato il *Random Forest*. Il numero di iterazioni previsto è 100 con 10 configurazioni iniziali identiche per i due modelli.

A questo scopo si è scelto di utilizzare il pacchetto **SMAC** per *Python*.

1 Esplorazione e preprocessing

Il *dataset* **fertility** contiene le analisi del seme di 100 volontari analizzato secondo i criteri per l'anno 2010 della *World Health Organization*. La concentrazione del seme è correlata a variabili socio-demografiche, elementi ambientali, stato di salute e stile di vita. Il *dataset* già in partenza risulta piuttosto ridotto per poter sfruttare a pieno il potere di classificazione di una rete neurale. Oltre a questo, la classe oggetto di classificazione presenta un forte sbilanciamento, come è possibile osservare in fig. 1. Questo *unbalancing* apre la strada ad una serie di problemi di validazione delle misure di accuratezza della classificazione

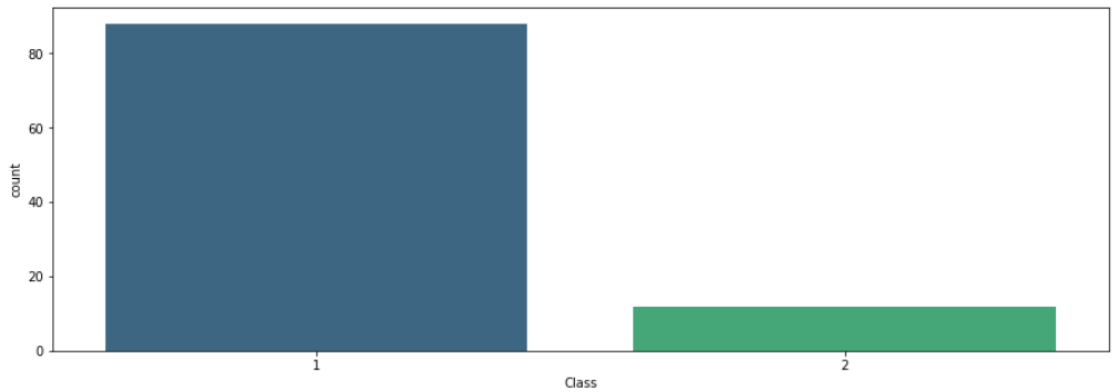

Figure 1: Distribuzione della classe all'interno del *dataset* **fertility**.

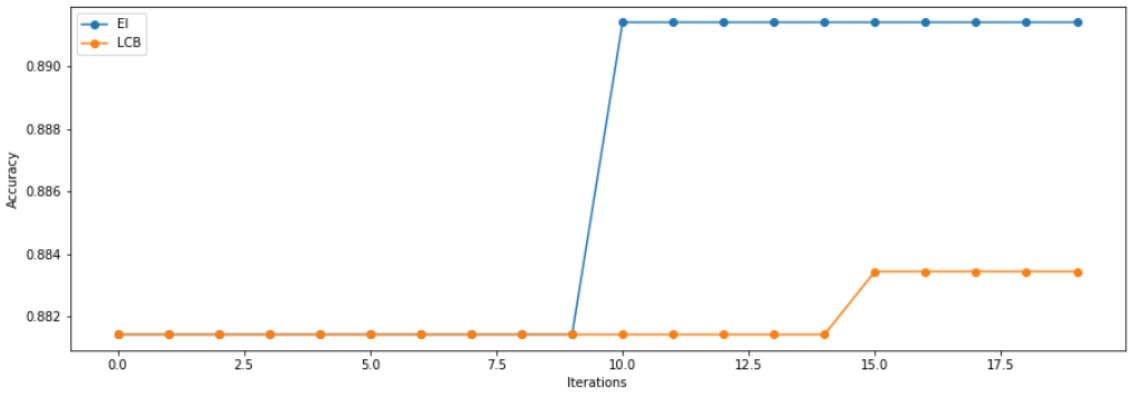

Figure 2: *Best seen per l'ottimizzazione dei modelli del primo step*.

dei modelli; lo scopo principale dell'*assignment* è però quello di implementare SMBO a livello tecnico e, per questa ragione, i problemi riguardanti il *dataset* sono trascurati. Posta questa importante premessa, per quanto riguarda le operazioni di *pre-processing*, il *dataset* si presenta già piuttosto pulito: l'unica operazione resasi necessaria è la conversione dell'etichetta della classe dall'originale formato [1, 2] al più *standard* [0, 1].

2 Modelli

I modelli realizzati sono sostanzialmente sei: quattro per il primo step, ovvero due SMBO, un *grid search* ed un *random search* e due SMBO per il secondo step.

Il primo step, come riportato sopra, consisteva essenzialmente nella realizzazione di due SMBO, con la possibilità di scegliere il modello surrogato da utilizzare tra il *Gaussian Process* ed il *Random Forest*: per comodità ed ordine di esecuzione si è scelto di utilizzare il *Gaussian Process* per entrambi i modelli. Una volta operata questa scelta, sostanzialmente **SMAC** richiede di definire la funzione obiettivo da ottimizzare, che nel nostro caso deve essere composta dalla definizione del modello con al suo interno l'indicazione degli iper-parametri da ottimizzare con in *output* lo *score* del modello stesso.

NB **SMAC** non permette di massimizzare lo *score* in *output* dalla funzione obiettivo, ma solo di minimizzare. Per questa ragione qualsiasi sia lo *score* di interesse, occorre esplicitarlo ottimizzandone la misura complementare: nel caso quindi dell'*accuracy* è necessario ottimizzare l'errore, ovvero

$$error = 1 - accuracy \tag{1}$$

Nel nostro caso inoltre l'*output* di questa funzione obiettivo non è una misura di valutazione *one-shot*, ma il risultato di una *10-fold cross validation*.

NB In particolare, la *10-folds cross validation* utilizzata è una *stratified cross validation*. Si è scelto di operare in questo modo per sopperire, anche solo in modo minimo, al forte sbilanciamento della classe oggetto di classificazione.

Tutti gli iper-parametri della rete neurale utilizzata, fatta eccezione ovviamente per quelli sottoposti ad ottimizzazione sono stati lasciati al loro valore di *default*, si è soltanto provveduto a determinare un *seed* sia per la rete neurale che per il meccanismo di *cross-validation* così che le 5 configurazioni iniziali fissate per il primo step e le 10 per il secondo non solo risultino uguali, ma vengano anche valutate identicamente. Fatte queste premesse, i risultati dei due modelli ottimizzati con le modalità previste dal primo step sono visibili in fig. 2. A parità quindi di configurazioni iniziali, di modello surrogato utilizzato, configurazione della rete neurale e numero di iterazioni, l'unica differenza tra i due modelli è rappresentata dalla funzione di acquisizione utilizzata: mentre il primo modello utilizza una *Expected Improvement* (EI) il secondo è

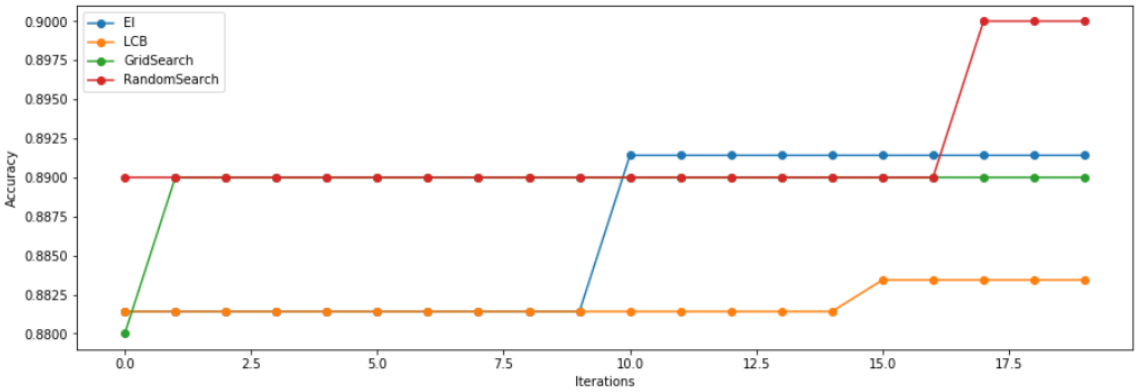

Figure 3: Best seen per l'ottimizzazione dei modelli del primo step includendo *Grid Search* e *Random Search*.

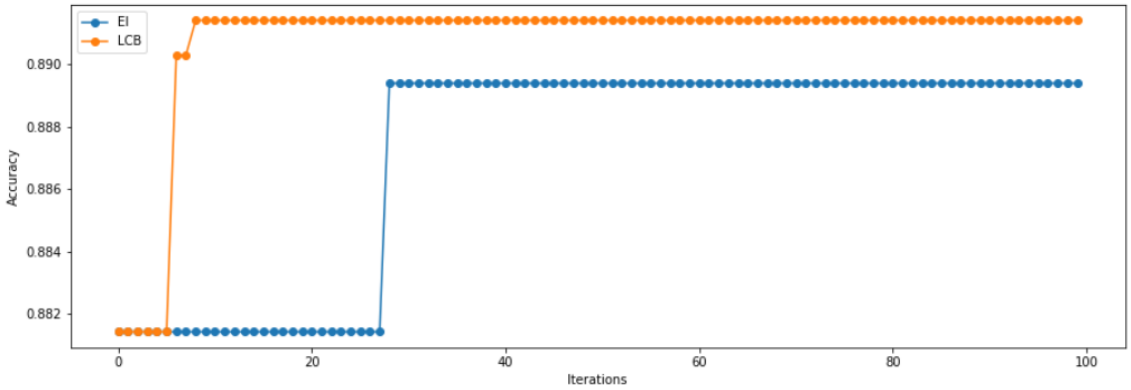

Figure 4: Best seen per l'ottimizzazione dei modelli del secondo step.

realizzato utilizzando un *Lower Confidence Bound* (LCB). La funzione di acquisizione determina la strategia di identificazione dei nuovi punti su cui valutare il modello, successivamente alla valutazione nei punti di configurazione iniziale definiti appunto *initial design*. La differenza più significativa tra le due funzioni di acquisizione consiste essenzialmente nel loro approccio: mentre la prima riesce a mantenere un buon bilanciamento tra *exploration* ed *exploitation* la seconda manifesta un atteggiamento più "ottimistico" nella scelta dei nuovi punti da utilizzare per valutare il modello.

Lo step uno oltre all'utilizzo del SMBO prevede, come detto in precedenza, anche l'utilizzo di 25 configurazioni di *Grid Search*, il meccanismo di ottimizzazione di iper-parametri più semplice ma anche computazionalmente meno efficiente, e 25 iterazioni di *Random Search*, un meccanismo meno guidato e che spesso raggiunge risultati piuttosto buoni. I risultati ottenuti da tutti e quattro i modelli realizzati per lo step uno sono visibili in fig. 3.

Il secondo step previsto dall'*assignment* differisce dal primo per pochi punti:

- come parametri da ottimizzare, oltre al *learning rate* ed il *momentum* vi è anche il numero di neuroni presenti nei due *hidden layer* in un *range* [1, 5];
- il numero di configurazioni iniziali è 10 anzichè 5;
- il numero di iterazioni è 100 invece di 20;
- per il modello surrogato non vi è più la possibilità di scegliere tra *Gaussian Process* e *Random Forest* ma è previsto di utilizzare il secondo a priori.

Anche per questo step rete neurale e meccanismo di *cross-validation* risultato identici a quelli utilizzati per il primo step in modo da mantenere una certa costanza, così come la scelta delle due funzioni di acquisizione da valutare, ovvero EI ed LCB. I risultati raggiunti dai due modelli ottimizzati per il secondo step sono visualizzabili in fig. 4.

3 Conclusioni

Le prestazioni di tutti i modelli sembrerebbero mediamente assestarsi sulla *zero-rule*, anche a causa del pesante sbilanciamento della classe oggetto di classificazione e, ovviamente, dello scarso numero di istanze presenti nel *dataset*: in questo senso alcune soluzioni applicabili potrebbero consistere nell'applicare tecniche di *data augmentation* oppure nell'utilizzare una misura di *scoring* pesata rispetto allo sbilanciamento della variabile *target*.

Dai grafici realizzati visualizzando il valore *best seen* appare chiaro come la funzione obiettivo risulti particolarmente piatta e, per questa ragione, l'ottimizzatore campioni i nuovi punti ad una distanza molto ridotta gli uni dagli altri, provocando così anche un appiattimento delle prestazioni ed uno scarso (o molto spesso del tutto nullo) miglioramento rispetto alle configurazioni iniziali definite a priori. Ciò vale sia per i modelli realizzati nel primo step che per quelli del secondo.

I risultati così ottenuti sono frutto di numerose inizializzazioni di diverse configurazioni iniziali, in modo che fosse possibile a tutti gli effetti apprezzare un miglioramento delle *performances* di classificazione durante il processo di ottimizzazione; molto spesso, durante i vari tentativi, accadeva infatti che già una delle configurazioni iniziali giungesse ad un minimo relativo e che l'ottimizzatore non riuscisse ad esplorare a sufficienza e fare in modo di migliorare ulteriormente lo *score* del modello. Per il caso particolare, inoltre, sembrerebbe che il modello di *Random Search* risulti essere il migliore, prima che per prestazioni innanzitutto per tempo di esecuzione. E' infatti vero che, come detto, questo metodo di ricerca risulti non ottimale rispetto alla *Sequence Model Based Optimization* ma, nel caso specifico, risulta molto rapido nello svolgere le 25 iterazioni richieste ed ottiene mediamente prestazioni migliori, anche se di pochissimo, per quanto concerne il primo step dell'*assignment*.

Per quanto riguarda il secondo step, invece, valgono sempre le stesse premesse fatte per il primo, ovvero che la funzione obiettivo risulti particolarmente piatta e l'ottimizzatore non riesca ad esplorare a sufficienza, anche utilizzando 100 iterazioni. Sarebbe interessante verificare le prestazioni dell'ottimizzatore utilizzando come criterio di stop il *runtime* invece del numero di iterazioni prefissato: in questo caso probabilmente avendo a disposizione sufficiente *budget* si otterrebbero risultati migliori in termini di classificazione, senza contare che il numero di iper-parametri sottoposti ad ottimizzazione risulta essere piuttosto ridotto. E' facile immaginare infatti che, se più iper-parametri fossero stati oggetto di ottimizzazione, le *performances* avrebbero ricevuto un significativo *improvement* (ovviamente a scapito del tempo richiesto) ma, come detto in principio, lo scopo dell'*assignment* verte prevalentemente sull'implementazione tecnica anzichè sul raggiungimento del massimo risultato.