Advanced Machine Learning - Assignment 5

Luca Gandolfi, matricola 807485

Dicembre 2019

1 Introduzione

L'assignment consiste nell'ottimizzare gli iperparametri di una rete neurale utilizzando Sequential Model based Optimization. Gli iperparametri ottimizati sono il Learning Rate, Threshold e numero di neuroni. In questo report verranno analizzati i risultati ottenuti nei due step richiesti e osservate le differenze rispetto ai metodi classici come Grid Search e Random Search.

2 Analisi preliminari

Il dataset utilizzato è Fertility: un problema di classificazione binaria su un insieme di dati relativamente contenuto, per l'esattezza 100 istanze, descritte da 9 feature numeriche più un attributo target. Il dataset è decisamente sbilanciato, presentando 88 record per la classe 1 e solo 12 record per la classe 2. Questo sbilanciamento andrà sicuramente ad influire sulle performance della rete neurale, portando a classificare molto bene le istanze della classe di maggioranza e presentando problemi per riconoscere la classe di minoranza.

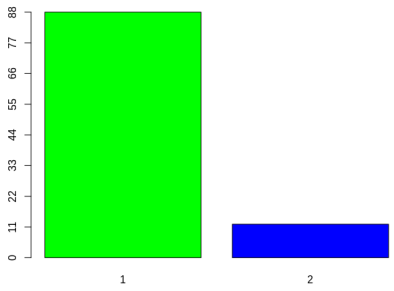

Figura 1: Distribuzione delle classi: 88 elementi per la classe 1 e 12 elementi per la classe 2.

3 Descrizione dell'esperimento

In questo paragrafo verranno descritti i due esperimenti svolti, spiegando le scelte prese in fase di modellazione e i risultati ottenuti. L'esperimento è stato svolto in linguaggio **R**, utilizzando **mlrMBO** come libreria per svolgere i task di ottimizzazione bayesiana. Il modello a cui ottimizzare gli iperparametri è una rete neurale con due hidden layers. La metrica utilizzata come riferimento per l'ottimizzazione è l'accuracy in 10-fold cross validation. In particolare, si è cercato di **massimizzare** tale valore.

Prima di svolgere i due Step dell'assignment, è stato necessario standardizzare le feature. In particolare, è stato scelto di sottrarre la media ad ogni valore e di dividere per la deviazione standard del corrispondente attributo, portando quindi la media a zero la deviazione standard pari a 1 per ogni feature.

3.1 Step 1

Gli iperparametri da ottimizzare nel primo esperimento sono il Learning Rate, limitato al range [0.01 ; 0.1], e il Threshold, limitato anch'esso al range [0.01 ; 0.1]. La rete neurale è stata definita tramite la libreria *neuralnet* con 4 neuroni nel primo hidden layer e 2 neuroni nel secondo. I restanti iperparametri sono stati lasciati con i valori di default.

L'esperimento è stato caratterizzato da un *initial design* nel quale sono stati scelti in modo random 5 punti sulla funzione obiettivo. L'initial design è stato tenuto fisso con due funzioni di acquisizione diverse: *Expected Improvement* e *Upper Confidence Bound*. Il budget restante dell'esperimento è stato limitato a sole 20 iterazioni del modello surrogato. Il modello surrogato utilizzato è *Kriging*, un metodo appartenente alla famiglia dei Gaussian Process, ovvero metodi di interpolazione modellati su processi Gaussiani.

3.1.1 Expected Improvement vs Upper Confidence Bound

Come anticipato, sono state testate due funzioni di acquisizione diverse, in particolare Exepected Improvement e Upper Confidence Bound. La scelta dell'ottimo è stata caratterizzata dalla tecnica del *Best Seen*, ovvero le configurazioni che hanno registrato il valore di accuracy più alta sono considerate come il punto ottimo, fino a che non si ottiene una nuova configurazione con risultati migliori.

Sono state effettuate diverse esecuzioni per valutare in media i risultati ottenuti dalle due funzioni di acquisizione, dato il non determinismo del modello surrogato. I grafici relativi all'andamento del Best Seen di tre esecuzioni sono mostrati di seguito. I grafici sono stati scelti per mostrare come i risultati dipendano non sono dall'initial design, ma anche da scelte non deterministiche prese in varie situazioni. Infatti, in Figura 2 è possibile osservare come UCB abbia ottenuto risultati migliori di EI, mentre in Figura 3 il contrario e infine, in Figura 4, i risultati siano stati praticamente identici.

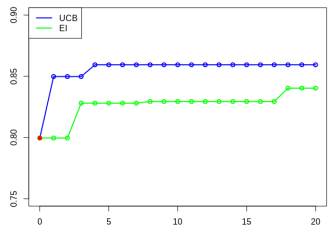

Figura 2: EI vs UCB: Esecuzione 1 - Best Seen.

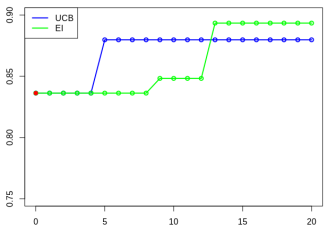

Figura 3: EI vs UCB: Esecuzione 2 - Best Seen.

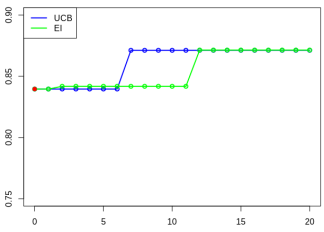

Figura 4: EI vs UCB: Esecuzione 3 - Best Seen.

L'asse delle ascisse indica il numero di iterazioni del modello surrogato, corrispondenti al budget definito precedentemente. L'iterazione più importante è l'ultima, ovvero la 20esima, che raccoglie il valore di Best Seen finale. L'asse delle ordinate indica l'accuracy (normalizzata in [0 ; 1]) ottenuta ottimizzando i parametri di Learning Rate e Threshold della rete neurale ed applicando una 10-fold Cross Validation stratificata, la quale mantiene le proporzioni delle classi del dataset originale anche nei 10 fold.

Il grafico è utile per comprendere con che facilità il modello è riuscito ad ottimizzare i due iperparametri. Per farlo, basta osservare la presenza di linee orizzontali, le quali indicano l'incapacità del modello di trovare nuove combinazioni che permettano di migliorare l'accuracy. Nel primo grafico ad esempio, UCB trova la sua configurazione migliore già dopo 5 iterazioni e non riesce a migliorarla per le successive 15.

In media, Expected Improvement ha riscontrato risultati superiori a Upper Confidence Bound. C'è da precisare però, che effettuare solo 20 iterazioni potrebbe essere riduttivo vista la dimensione del dataset e la profondità della rete, entrambe molto contenute. Eseguendo un numero maggiore di iterazioni i risultati finali potrebbero essere diversi.

I risultati di queste tre esecuzioni a confronto sono mostrati in Tabella 1. Come è possibile notare, il valore più alto di accuracy in 10-fold Cross Validation è stato raggiunto utilizzando come funzione di acquisizione Expected Improvement, mentre gli altri valori sono abbastanza sotto le aspettative. Considerando la distribuzione delle classi, ci si aspetta che la rete neurale riesca a riconoscere nella maggior parte dei casi gli esempi della classe di maggioranza e sbagli molto a riconoscere gli esempi della classe di minoranza. Seguendo questa logica, è possibile ipotizzare valori di accuracy tendenti allo 0.88, dato che le istanze della classe di maggioranza sono 88. Provando a spiegare i valori ottenuti, è possibile confermare quello che è stato già detto a proposito del numero di iterazioni, ovvero che probabilmente siano troppo poche per ottenere i risultati migliori. Anche la dimensione della rete, con il numero fisso di neuroni potrebbe aver inciso sulle performance.

Acquisition Function	Esperimento 1	Esperimento 2	Esperimento 3
Upper Confidence Bound	0.859394	0.879798	0.871212
Expected Improvement	0.840303	0.8934343	0.871414

Tabella 1: Accuracy in 10-fold Cross Validation nei tre esperimenti.

In genere, i risultati migliori sono stati ottenuti con Learning Rate e Threshold molto bassi, di poco maggiori di 0.01.

Dopo numerosi test, il valore di accuracy più alto registrato è quello ottenuto con l'esperimento 2. I relativi valori degli iperparametri ottimizzati sono descritti in Tabella 2. E' necessario rimarcare che il risultato è stato ottenuto con Expected Improvement.

Accuracy	Learning Rate	Threshold
0.8934343	0.06004727	0.03470116

Tabella 2: Valori degli iperparametri nella migliore configurazione trovata.

Dato che gli iperparametri da ottimizzare sono soltanto due, è possibile rappresentare le configurazioni testate dal Sequential Model Based Optimization in una griglia bi-dimensionale, rappresentata da due assi cartesiani: l'asse delle ascisse rappresenta i valori di Learning Rate e l'asse delle ordinate i valori di Threshold.

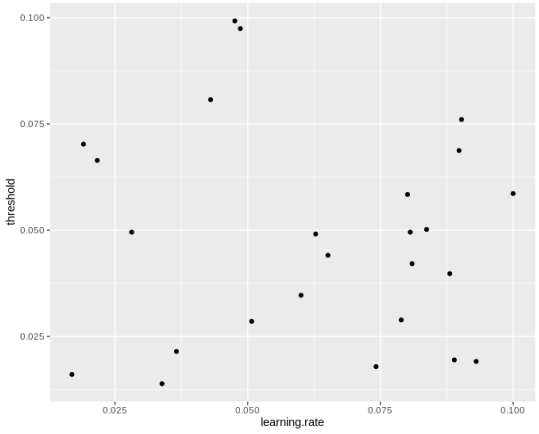

Figura 5: Rappresentazione bi-dimensionale delle combinazioni testate da **Expected Improvement** nell'esperimento 2.

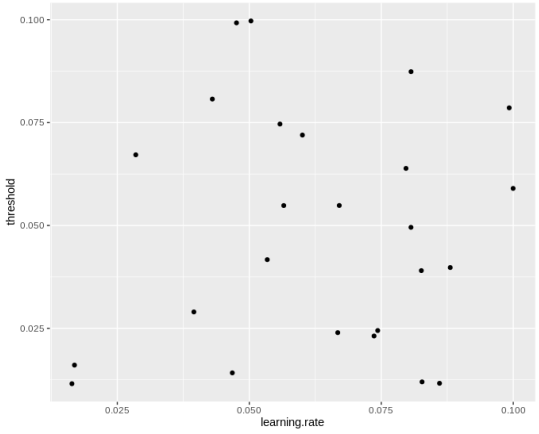

Figura 6: Rappresentazione bi-dimensionale delle combinazioni testate da **Upper Confidence Bound** nell'esperimento 2.

Come è possibile osservare in Figura 5, Expected Improvement si è concentrato molto in zone con Threshold basso, esplorando diverse configurazioni in zone concentrate prima di procedere in una

nuova zona. Le zone isolate con Threshold alto indicano che il modello si è reso conto di essere lontano dalla configurazione migliore. Idealmente, potrebbe essere possibile tagliare il piano in due triangoli rettangoli tramite una retta obliqua passante per i due angoli opposti in basso a sinistra ed in alto a destra. Questo taglio permette di dividere al meglio la zona maggiormente considerata dalla funzione di acquisizione e la zona composta da componenti isolate.

Upper Confidence Bound, osservabile in Figura 6, ha coperto di più lo spazio dei valori possibili, sinonimo probabilmente di indecisione come dimostrato dal risultato ottenuto, più basso rispetto a Expected Improvement. Le combinazioni sono comunque per lo più concentrate nello spazio di piano delimitato dal Learning Rate maggiore di 0.5.

Il limitato budget ha impedito all'algoritmo di testare più configurazioni vicine alle zone ottime e di coprire meglio lo spazio. Questo discorso vale per entrambe le funzioni di acquisizione testate.

3.1.2 Grid Search vs Random Search

La libreria mlrMBO offre la possibilità di definire dei design classici, come Grid Design e Random Design. La loro esecuzione permette di eseguire, in modo abbastanza inefficiente, una ottimizzazione degli iperparametri. Anche in questo caso, avere soltanto due iperparametri da ottimizzare rende possibile il rappresentare le configurazioni provate in uno spazio bi-dimensionale. Non è necessario in questo caso un initial design, il budget disponibile è di 25: una griglia 5x5 per il Grid Search e 25 configurazioni random per il Random Search.

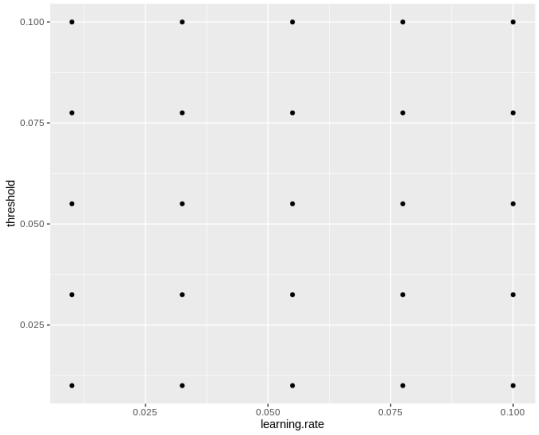

Figura 7: Grid Search design.

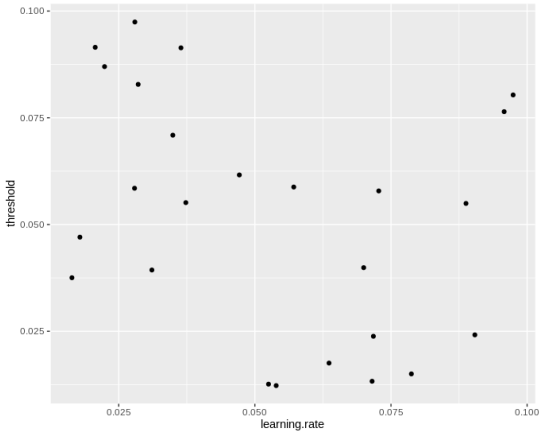

Figura 8: Random Search design.

Figura 7 e Figura 8 mostrano la copertura dello spazio dei valori di Learning Rate e Threshold definiti. Come era ipotizzabile, è possibile notare come Grid Search e Random Search non coprano bene lo spazio con sole 25 configurazioni, soprattutto quest'ultima presenta dei grossi vuoti per lo più in zone con Learning Rate basso e in zone con Threshold Alto.

Design	Accuracy	Learning Rate	Threshold
Grid Search	0.850707	0.0775	0.0775
Random Search	0.840303	0.0164256	0.03753649

Tabella 3: Valori degli iperparametri nella migliore configurazione di Grid Search e Random Search.

Osservando i valori ottenuti è possibile fare delle considerazioni. Come prima cosa, i valori degli iperparametri sono ben lontani dai punti che hanno permesso di raggiungere 0.89 di accuracy. Random Search è riuscito ad avvicinarsi all'intervallo dei valori di Threshold corretto (supponendo sia l'intorno di valori di 0.03), nonostante poi provare soltanto 3 configurazioni di Learning Rate diverse in questo intervallo e senza mai avvicinarsi a valori alti (e corretti). Grid Search invece, si è avvicinato all'intervallo di valori corretto per il Learning Rate (supponendo sia l'intorno di

0.06), sbagliando strada per il Threshold. E' bene considerare però, che i valori di accuracy più alti registrati siano in realtà delle anomalie rispetto alla media dei risultati ottenuti, i quali si aggiravano spesso intorno allo 0.86 di accuracy. Le performance quindi non sono state poi così male. Come seconda considerazione, è possibile osservare come anche cambiamenti significativi dei valori degli iperparametri portino in realtà modifiche minime sui risultati. Questo potrebbe essere dovuto a: problema di classificazione sbilanciato e allo stesso tempo troppo semplice (9 feature per 100 record), e l'utilizzo di una rete neurale multistrato, che è un modello di base molto performante. Infine, la terza considerazione possibile riguarda il grosso limite di questi due approcci base: la mancanza di memoria delle configurazioni precedentemente testate (e quindi dei valori ottimi riscontrati) da parte di Random Search e il vincolo di una griglia di combinazioni fisse da seguire da parte di Grid Search. Quest'ultimo problema infatti è vincolante nel caso in cui i valori ottimi si trovino proprio a metà tra due configurazioni (come in questo caso). Problema che potrebbe essere ovviamente risolto aumentando il budget ma non sempre questo è disponibile in grandi quantità. Anche Grid Search non possiede memoria, la quale potrebbe essere d'aiuto per testare prima le combinazioni vicine al valore più alto trovato, sempre rispettando i punti della griglia.

In Figura 9 viene mostrato l'andamento della Best Seen accuracy nel corso dei vari tentativi. In questo caso non è presente un Initial Design comune.

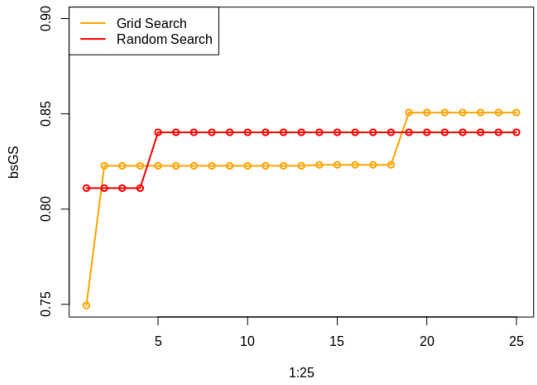

Figura 9: Andamento delle accuracy in 10-fold CV a confronto tra Random e Grid Search.

3.2 Step 2

Gli iperparametri da ottimizzare nel secondo esperimento sono il Learning Rate, limitato al range [0,01 ; 0.1], il Threshold, limitato anch'esso al range [0.01 ; 0.1], e il numero di neuroni dei due hidden layer, limitato a valori interi tra 1 e 5. Non vi sono differenze sostanziali con il primo esperimento se non per i due iperparametri aggiuntivi da ottimizzare, per un totale di 4 iperparametri, e il budget: 10 punti random per *initial design* più 100 iterazioni come budget. Anche in questo caso l'initial design è stato tenuto fisso con due funzioni di acquisizione diverse: *Expected Improvement* e *Upper Confidence Bound*.

Data la presenza sia di iperparametri continui che discreti, in questo caso è stato necessario utilizzare *RandomForest* come modello surrogato, poiché è leggermente più performante di GP con iperparametri continui e discreti insieme.

3.2.1 Expected Improvement vs Upper Confidence Bound

Come per lo Step 1, sono state confrontate le performance con due funzioni di acquisizione diverse: Expected Improvement e Upper Confidence Bound, partendo dalla stessa configurazione iniziale per avere uniformità nei risultati. I risultati sono stati ottenuti applicando una Cross Validation

Stratificata su 3 Fold. Le ragioni dietro questa scelta sono motivate nel paragrafo successivo 3.2.2 - Problematiche incontrate.

La richiesta di ottimizzare due iperparametri in più rispetto allo Step 1 ha complicato il processo di ottimizzazione, vedendo necessario l'utilizzo di un budget maggiore. Nonostante il budget fissato a 100 iterazioni, i risultati si sono rilevati in media abbastanza deludenti: soltanto una volta su numerosi test è stato possibile ottenere 0.89 di accuracy.

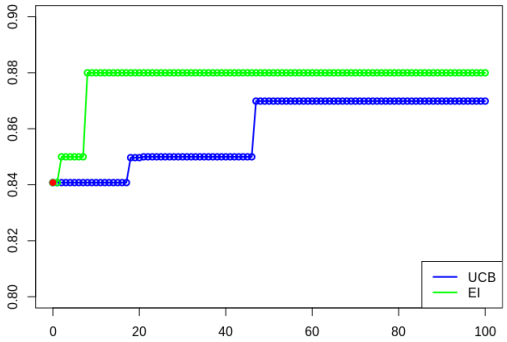

Figura 10: EI vs UCB con Best Seen dopo 100 iterazioni.

Anche in questo esperimento, Expected Improvement ha riscontrato, in media, performance migliori rispetto a Upper Confidence Bound. In Figura 10 ad esempio, è possibile osservare come EI raggiunga molto presto il suo Best Seen e lo mantenga fino all'ultima iterazione del budget disponibile. UCB, invece, pur effettuando lo stesso numero di aggiornamenti del Best Seen (è possibile osservare 2 "salite" nel grafico), non ha trovato la stessa configurazione.

Acquisition Function	Accuracy	Learning Rate	Threshold	Hidden Layer 1	Hidden Layer 2
Upper Confidence Bound	0.8698752	0.06337156	0.07110823	1	2
Expected Improvement	0.8799762	0.02199106	0.04808323	4	1

Tabella 4: Accuracy in 3-fold Cross Validation e Iperparametri ottimizzati dopo 100 iterazioni.

Analizzando i valori degli iperparametri nelle configurazioni migliori, è possibile notare come UCB abbia utilizzato un Learning Rate più alto, lontano dal risultato migliore ottenuto nello Step 1, mentre EI si sia avvicinato di più. Per quanto riguarda i valori di Threshold invece, questi sono molto bassi in entrambi i casi. I risultati, tuttavia, non sono molto confrontabili con lo Step 1 visto il numero di neuroni trovati.

Limitandosi al confronto tra le due funzioni di acquisizione, gli iperparametri ottenuti sono tra loro molto diversi, dimostrando gli approcci differenti adottati dalle due funzioni.

Considerando invece il budget, probabilmente utilizzare valori maggiori avrebbe portato ad avvicinarsi di più all'ottimo. Tuttavia, non è stato possibile sperimentare a causa di alcune problematiche occorse.

3.2.2 Problematiche incontrate

La problematica principale occorsa durante lo svolgimento dello Step2, riguarda il crash del processo di Cross Validation. Utilizzando Kriging come Gaussian Process, questo problema si è verificato molto di rado, tuttavia, utilizzando RandomForest la frequenza è aumentata. L'utilizzo

di un budget di 100 iterazioni (oltre alle 10 dell'initial design) aumenta in modo importante la probabilità che si verifichi il crash.

Come prima soluzione è stato pensato di diminuire il numero di fold nella Cross Validation, per permettere alla classe di minoranza di avere più istanze in ogni fold. Sono stati provati 5 fold e 3 fold. Con quest'ultima configurazione la probabilità di crash si è abbassata, ma non si è azzerata, posticipando spesso il problema dopo circa 40 iterazioni.

Tramite un workaround, eseguendo manualmente 100 iterazioni e costruendo il design dopo ognuna di queste, è stato possibile ottenere qualche esecuzione che sfruttasse il budget per intero, come mostrato in Figura 10, nel paragrafo precedente.

In Figura 11 viene mostrato un esempio di esecuzione in cui, utilizzando Expected Improvement, la 3-fold Cross Validation non è riuscita a terminare, riscontando problemi poco prima di 50 iterazioni. Come è possibile osservare però, fino a quel punto era stato ottenuta un'accuracy in Best Seen molto alta, precisamente 0.8897802.

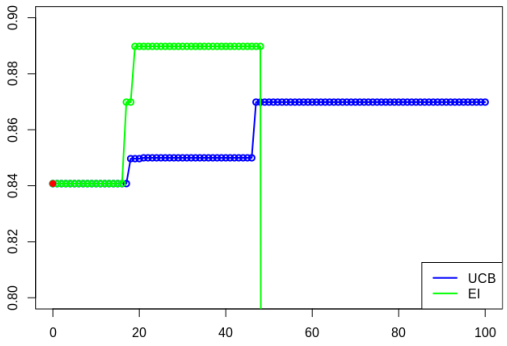

Figura 11: Esecuzione in cui la CV con Expected Improvement è andata in crash.

Paradossalmente, è stato riscontrato che quando la Cross Validation andava in crash, l'accuracy in Best Seen era maggiore rispetto a quando l'esecuzione terminava senza problemi.

4 Conclusioni

L'ottimizzazione degli iperparametri è sicuramente un task fondamentale per ottenere buoni risultati senza perdere tempo a provare manualmente tante configurazioni diverse. Ovviamente, è necessaria una conoscenza di base per permettere di restringere i range di valori possibili per gli iperparametri da ottimizzare, altrimenti il processo potrebbe impiegare troppo tempo.

Lavorando con budget bassi, è doveroso accontentarsi dei risultati ottenuti, pur sapendo che probabilmente il valore ottimo della funzione obiettivo non è ancora stato trovato. I risultati ottenuti nei due esperimenti mostrano come questa affermazione sia veritiera: solo in pochissime esecuzioni è stato possibile raggiungere 0.889 e 0.89 di accuracy, probabilmente valori vicinissimi all'ottimo data la distribuzione delle classi nel dataset. La media dei risultati infatti, anche utilizzando budget 100 (possibile grazie alla piccola dimensione del dataset), si è rivelata essere sui 0.86/0.87. Differenze che non sono così banali, poichè in questo caso molto probabilmente viene sbagliato anche qualche esempio della classe di maggioranza.

Un'altra componente importante riscontrata che influisce sui risultati è il non determinismo. Questo è dovuto in primo luogo a diversi initial design utilizzati in diverse esecuzioni e, in secondo luogo, alla componente probabilistica del Sequential Model Based Optimization, che spesso permette di prendere path diversi pur convergendo alla stessa soluzione di ottimo (locale o globale) dopo sufficiente budget.

Il problema principale di Grid Search e di Random Search è la totale ignoranza verso le combinazioni precedentemente testate, questo spesso risulta in una perdita di tempo a testare configurazioni ben lontane dai valori ottimi. Negli esperimenti svolti, Grid Search è riuscito a coprire meglio lo spazio degli iperparametri, ottendo infatti risultati migliori rispetto a Random che, generalmente, va a fortuna.

Gli Step 1 e 2 non sono molto confrontabili tra loro per via dei due diversi modelli surrogati adottati e per il numero di iperparametri da ottimizzare. Tuttavia, risulta evidente come un aumento del numero di questi ultimi comporti un significativo aumento di budget richiesto per raggiungere buoni risultati. I risultati più alti sono stati ottenuti infatti nello Step 1, nonostante il budget molto economico adottato. Solo in un esperimento nello Step 2 è stato possibile raggiungere 0.89 di accuracy, ma in 3-fold Cross Validation e con Expected Improvement, ma con crash successivo su Upper Confidence Bound.

Il problema riguardante il crash della Cross Validation ha influito molto sui risultati dello Step 2, non potendo fare delle analisi accurate poichè i risultati migliori sono stati ottenuti proprio quando il modello andava in crash. Questo problema potrebbe essere dovuto principalmente dalla distribuzione delle classi del dataset (fortemente sbilanciato) e l'opzione di stratificazione della Cross Validation non ha comunque aiutato.