Optimization Methods
Hyperparameters Optimization

emanuele.vivoli@stud.unifi.it
federico.magnolfi2@stud.unifi.it

Febbraio 2019

1 Introduzione

L'obiettivo di questo elaborato è quello di applicare e confrontare due diversi metodi di ottimizzazione globale per funzioni costose, che effettuano una ricerca globale per la scelta di iperparametri nell'addestramento di una rete neurale.

2 Descrizione del problema

L'addestramento di una rete neurale consiste nell'ottimizzare una funzione obiettivo detta loss, che misura quanto le predizioni della rete differiscono dalle uscite desiderate, in funzione di un certo numero di parametri ed iperparametri.

I parametri (o pesi) possono essere appresi durante l'addestramento sfruttando il gradiente della loss rispetto ad essi (metodo del gradiente e derivati), oppure sfruttando un'approssimazione della funzione al secondo ordine (metodi quasi-Newton). Queste tecniche non sono però utilizzabili per la scelta degli iperparametri, che devono essere già scelti prima dell'inizio della fase di allenamento.

La scelta degli iperparametri può essere formulata come un problema di ottimizzazione globale in cui valutare la funzione obiettivo è molto costoso (necessita infatti di un addestramento della rete), e per la quale non si ha un metodo efficiente per il calcolo del gradiente o della matrice Hessiana. In questo studio si considera un addestramento di una rete relativamente semplice per confrontare due algoritmi di ottimizzazione globale, entrambi basati sul valutare un modello surrogato della funzione obiettivo: per la creazione del modello, il primo algoritmo usa l'interpolazione di funzioni di base radiali (RBF), mentre il secondo usa l'inferenza Bayesiana e i processi stocastici gaussiani.

(a) airplane (b) airplane (c) airplane

(d) automobile (e) automobile (f) automobile

(g) deer (h) deer (i) deer

Figure 1: Esempi di immagini del dataset CIFAR-10

3 Descrizione dell'esperimento

Si è deciso di utilizzare una rete neurale convoluzionale non molto profonda per classificare le immagini del dataset CIFAR-10. Gli iperparametri da ottimizzare individuati sono il learning-rate e la penalità dei regolarizzatori. Come libreria di Machine Learning si è usato PyTorch, mentre gli ottimizzatori di iperparametri utilizzati sono rbfopt e bayesian-optimization, il tutto nel linguaggio Python 3.

3.1 Dataset

Il dataset CIFAR-10 è composto da 60,000 immagini a colori di dimensione 32x32, suddivise in 10 classi. In Figure 1 si mostra un esempio di immagini che compongono il dataset.

3.2 Rete neurale

La rete neurale usata negli esperimenti ha 9 livelli e circa 120000 parametri. Di solito per fare gli esperimenti sulla classificazione di immagini vengono usate

reti più profonde e con molti parametri in più. Si è deciso di usare una rete relativamente semplice in quanto in questo esperimento non si cerca di raggiungere un nuovo record di accuratezza, ma si vuole invece valutare la bontà di due algoritmi di valutazioni di iperparametri. La rete utilizzata è la seguente:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 16, 30, 30]             448
            Conv2d-2          [-1, 16, 28, 28]           2,320
         MaxPool2d-3          [-1, 16, 14, 14]               0
            Conv2d-4          [-1, 32, 12, 12]           4,640
            Conv2d-5          [-1, 32, 10, 10]           9,248
         MaxPool2d-6            [-1, 32, 5, 5]               0
            Linear-7                 [-1, 120]          96,120
            Linear-8                  [-1, 84]          10,164
            Linear-9                  [-1, 10]             850
================================================================
Total params: 123,790
```

3.3 Algoritmi

Gli algoritmi utilizzati sono Radial Base Function e Bayesian Optimization, dei quali sono state usate due implementazioni presenti su GitHub.

L'implementazione dell'algoritmo basato su RBF fornisce principalmente tre classi: una per specificare il numero di iperparametri ed il loro dominio, una seconda per specificare il numero di valutazioni che si vogliono fare, e una terza classe utilizzata per eseguire il vero e proprio algoritmo di ottimizzazione. I primi passi della ricerca sono scelti in modo casuale, ed è compito dell'ottimizzatore decidere il numero esatto dei passi random prima che entri in gioco l'algoritmo vero e proprio.

L'algoritmo Bayesiano fornisce due soli classi che si dividono i ruoli di specificare le impostazioni della black-box e strutturare l'algoritmo di ottimizzazione. Anche per questo algoritmo i primi passi della ricerca sono di esplorazione globale randomica, concedendo però la possibilità di specificare il numero preciso di iterazioni per la ricerca casuale.

Un ultima differenza tra le implementazioni usate di rbfopt e BayesianOptimization è che la prima minimizza la funzione obbiettivo, mentre la seconda la massimizza. Per questo caso d'uso la funzione obiettivo scelta è la loss del validation set, e dunque è risultato necessario invertirla per l'ottimizzazione con BayesianOptimization, cambiandola di segno.

3.4 Esperimenti

Gli esperimenti sono stati impostati in modo da avere una ricerca casuale iniziale di 5 iterazioni precise per l'ottimizzatore Bayesiano, e di massimo 5 iterazioni

per RBF. Dopo le scelte casuali iniziali, gli algoritmi iniziano a scegliere i nuovi punti da valutare ognuno in base al proprio criterio; il numero massimo di valutazioni complessive è stato impostato a 25 per entrambi gli algoritmi. Ad ogni scelta della coppia learning rate e weight decay viene attuato un addestramento della rete su 70 epoche; in tutti gli allenamenti i pesi della rete vengono inizializzati agli stessi valori, in modo da evitare che inizializzazioni diverse influenzino i risultati degli allenamenti. La tecnica di early stopping non è stata utilizzata, ma per la valutazione della funzione obiettivo (validation loss) si è considerato il miglior valore assunto da essa. Per ogni epoca, si sono salvati i valori di loss e accuratezza sia sul train che sul validation, in modo da visualizzarli tramite tensorboard. Per ogni addestramento, si salvano su un file csv informazioni relative al valore degli iperparametri e alle loss/accuratezze su train/validation/test: per ogni metrica si riporta soltanto il valore misurato alla fine dell'epoca corrispondente alla migliore loss sul validation set.

4 Struttura del codice

Il progetto è composto principalmente da 3 file:

- **utils.py**: contiene metodi per il download del dataset CIFAR-10 e per la suddivisione del dataset in train, validation, test;
- **net.py**: contiene una classe che definisce la struttura della rete, permette di allenarla e di valutarla;
- **evaluate.py**: script principale, contiene la creazione degli ottimizzatori (RBF-opt e Bayesian-opt) e la funzione di evaluate (valutazione della funzione obiettivo).

Il codice sviluppato è interamente disponibile sulla piattaforma GitHub (link al repository).

5 Risultati

L'algoritmo rbfopt trova delle combinazioni dei due iperparametri che portano il valore della funzione obbiettivo sotto l'unità, con accuratezze che superano il 68%. BayesianOptimization ha risultati peggiori, sia in termini di loss che di accuratezza (con un massimo di 64%). In entrambi i casi, i valori di accuratezza sono lontani dai record raggiunti su questo dataset (94% nel 2011; 96.53% nel 2015; link), ma raggiungere questi valori non era l'obiettivo dell'esperimento.

In tabella 1 e in tabella 2 sono riportati i risultati numerici che sono stati ottenuti rispettivamente per l'algorimto basato su RBF e per l'algoritmo bayesiano.

6 Conclusioni

Entrambi gli algoritmi testati raggiungono un risultato simile, ma l'algoritmo basato su RBF si è rivelato migliore sia in termini del valore della funzione obiettivo (validation loss) che in termini di accuratezza.

I metodi basati sulle funzioni radiali hanno necessità di meno iterazioni casuali iniziali per cominciare a dare dei risultati considerevoli, ma perdono molto nella ricerca globale. Al contrario, i metodi basati su inferenze bayesiane necessitano di più iterazioni random iniziali. Dato che ogni valutazione della funzione obiettivo costa molto, è infatti un problema di ottimizzazione di funzioni costose, si potrebbe preferire il metodo rbf al metodo bayesiano.

opt	lRate	wDecay	trLoss	vaLoss	teLoss	trAcc	vaAcc	teAcc
RBF	0.0915	0.0089	5.4735	2.3051	2.3073	0.099	0.1011	0.1
RBF	0.0592	0.0001	2.3666	2.3039	2.3041	0.0992	0.1011	0.1
RBF	0.0218	0.0058	2.3048	2.3035	2.304	0.0985	0.0952	0.1
RBF	0.0001	0.0001	0.8364	1.0608	1.0525	0.7094	0.6406	0.6363
RBF	0.0002	0.0025	0.7716	1.0506	1.7627	0.7324	0.6521	0.6201
RBF	0.0001	0.0013	0.7956	1.0267	1.1574	0.7214	0.6478	0.64
RBF	2.5E-05	0.0019	1.258	1.3152	1.3058	0.5493	0.5318	0.5285
RBF	0.0001	0.0009	0.9784	1.1156	1.1068	0.6588	0.6118	0.6069
RBF	1.0E-05	0.0029	1.4797	1.5095	1.4882	0.4657	0.4555	0.4599
RBF	0.0673	0.0048	2.3105	2.3035	2.3042	0.1012	0.1011	0.1
RBF	0.0349	0.0026	2.3091	2.3037	2.3043	0.0988	0.1011	0.1
RBF	0.0239	3.4E-06	2.3052	2.3039	2.3041	0.0978	0.0952	0.1
RBF	0.004	0.0024	0.8793	0.9442	0.9911	0.689	0.6704	0.6481
RBF	0.0041	0.0024	0.8889	0.9523	0.987	0.6845	0.6658	0.6506
RBF	0.0038	0.0024	0.8716	0.9432	1.0121	0.6911	0.6649	0.6461
RBF	0.0998	0.0018	9.8161	2.3055	5.9267	0.0984	0.1011	0.0999
RBF	0.0517	0.0085	2.3075	2.3035	2.3039	0.0997	0.1011	0.1
RBF	0.0141	0.0027	1.6079	1.5665	1.5891	0.4022	0.4251	0.408
RBF	0.005	0.0025	0.9634	0.9939	1.0143	0.6598	0.6473	0.6412
RBF	0.0037	0.0024	0.8394	0.8985	0.9147	0.7046	0.6863	0.6818
RBF	0.003	0.0024	0.7988	0.9398	0.9932	0.7207	0.6721	0.6619
RBF	0.0034	0.0024	0.8106	0.9204	0.9586	0.7164	0.6782	0.669
RBF	0.0039	0.0024	0.8945	0.9751	1.0406	0.6811	0.654	0.6367
RBF	0.0037	0.0024	0.8625	0.9235	0.9771	0.6967	0.6765	0.6636
RBF	0.0007	0.01	0.8405	0.9577	0.9668	0.7081	0.6619	0.6625

Table 1: Risultati ottenuti con gli esperimenti con l'algoritmo RBF. Ogni riga della tabella corrisponde ad un allenamento con conseguente valutazione della funzione obiettivo; nelle diverse colonne sono riportati i valori degli iperparametri e delle metriche. In giallo è evidenziata la colonna della funzione obiettivo, in verde è evidenziato il risultato migliore

opt	lRate	wDecay	trLoss	vaLoss	teLoss	trAcc	vaAcc	teAcc
BAY	0.0662	0.0022	4.1841	2.3041	2.3045	0.1001	0.1011	0.1
BAY	0.0182	0.0084	1.7597	1.7327	1.9156	0.3314	0.3416	0.272
BAY	0.0643	0.0046	13.7495	2.3034	2.3041	0.098	0.1011	0.1
BAY	0.0279	0.0076	2.3048	2.3033	2.3039	0.0984	0.1011	0.1
BAY	0.0008	0.0035	0.9235	1.0202	1.3873	0.6754	0.6427	0.637
BAY	0.1	0.01	4.7634	2.2938	2.3082	0.1025	0.1067	0.1
BAY	1.0E-05	0.01	1.5378	1.5682	1.5432	0.4479	0.4395	0.4458
BAY	0.0097	0.0E+00	2.3036	2.3036	2.3033	0.0976	0.0952	0.1
BAY	0.0854	0.0E+00	8.1662	2.3063	2.3064	0.0995	0.1011	0.1
BAY	0.0464	0.0E+00	2.4958	2.3042	2.3039	0.0985	0.0952	0.1
BAY	0.0778	0.01	3.6174	2.3037	2.3054	0.0978	0.1011	0.1
BAY	1.0E-05	0	1.4389	1.4685	1.4525	0.481	0.4734	0.4726
BAY	0.0087	0.01	2.3035	2.3033	2.3031	0.0996	0.0952	0.1
BAY	0.0506	0.01	2.2847	2.2197	2.3037	0.1386	0.1539	0.1
BAY	0.1	0.0E+00	17.6752	2.3078	2.3084	0.1017	0.1011	0.1
BAY	0.0226	0.0E+00	2.3095	2.3045	2.3041	0.0985	0.0952	0.1
BAY	0.0396	0.01	2.3077	2.303	2.3038	0.0984	0.1002	0.1
BAY	0.0892	0.01	3.1804	2.3044	2.3068	0.1001	0.1011	0.1
BAY	1.0E-05	0.0053	1.5063	1.536	1.5105	0.4576	0.4535	0.454
BAY	0.0352	0.0E+00	2.3524	2.3044	2.304	0.098	0.0952	0.1
BAY	0.0563	0.0E+00	2.835	2.3043	2.3041	0.0994	0.1011	0.1
BAY	0.0757	0.0E+00	2.3097	2.3054	2.3052	0.0994	0.1011	0.1
BAY	0.0045	0.0034	0.9925	0.9928	1.0023	0.6489	0.645	0.6471
BAY	0.0693	0.01	2.9753	2.3037	2.3037	0.0975	0.1011	0.1
BAY	0.0585	0.01	6.9939	2.3035	2.3038	0.0978	0.0952	0.1

Table 2: Risultati ottenuti con gli esperimenti con l'algoritmo bayesiano. Ogni riga della tabella corrisponde ad un allenamento con conseguente valutazione della funzione obiettivo; nelle diverse colonne sono riportati i valori degli iperparametri e delle metriche. In giallo è evidenziata la colonna della funzione obiettivo, in verde è evidenziato il risultato migliore

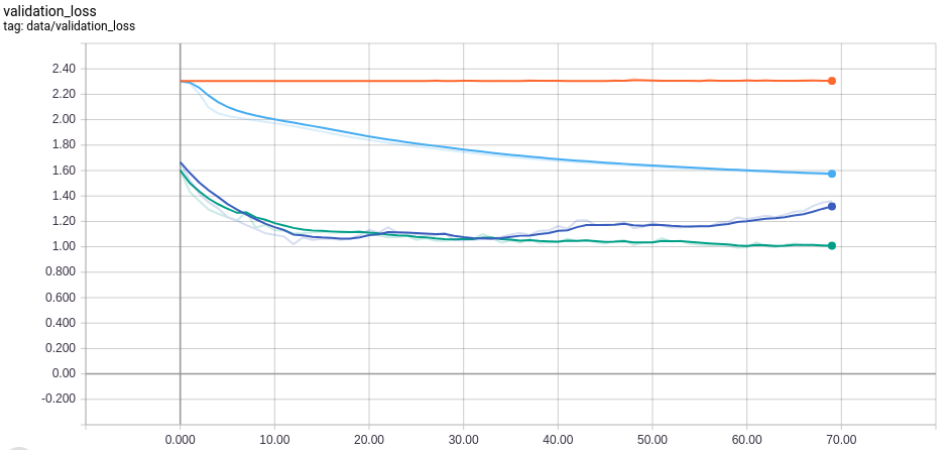

Figure 2: Validation loss con ottimizzatore Bayesiano

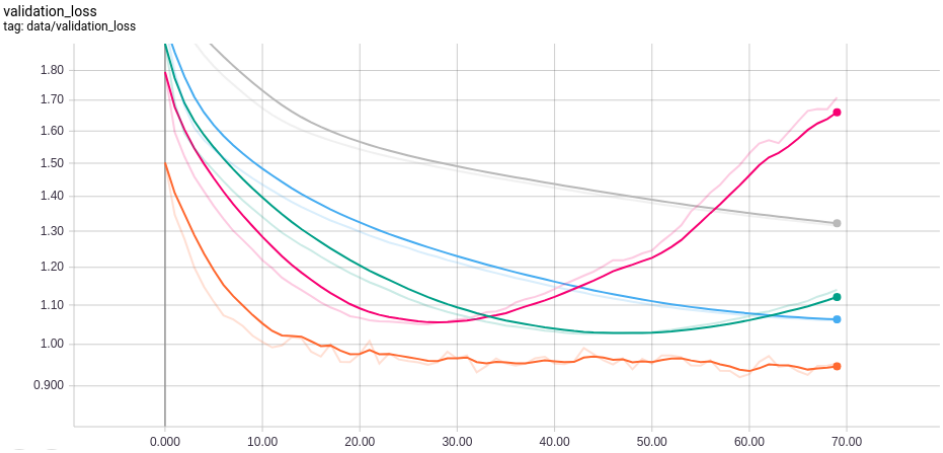

Figure 3: Validation loss con ottimizzatore RBF