Università di Pisa

ML 2018 - Report

Andrea Pedrotti
matricola: 569692

Tiziano Labruna
matricola: 516373

10 gennaio 2019

Progetto A

Sommario

Implementazione di Rete Neurale Artificiale fully connected in Python. Analisi della convergenza in fase di allenamento per l'individuazione dei migliori iper-parametri tramite K-Fold Cross Validation, successivo test su task di classificazione (MONK) e regressione (MLCUP).

1 Introduzione

L'obiettivo del progetto è stato quello di implementare in Python una rete neurale artificiale in grado di risolvere i task indicati durante il corso. La rete *fully connected* è stata allenata tramite algoritmo di *backpropagation* e *online gradient descent* con possibilità di utilizzo di *momentum*, *step decay* e regolarizzazione L2. Si sono poi valutati i miglioramenti ottenibili applicando le strategie di inizializzazione dei pesi proposte da Glorot-Bengio e He-Zhang-Ren-Sun.

2 Metodo

2.1 Librerie utilizzate e scelte di implementazione

La rete è stata implementata utilizzando il linguaggio di programmazione Python 3. Si sono utilizzate le librerie NumPy - per velocizzare i tempi di computazione attraverso il calcolo matriciale - e MatPlotLib per la produzione dei grafici sugli andamenti in fase di *training* e *testing*.
Si è cercato di mantenere un approccio modulare e orientato agli oggetti in modo da poter istanziare differenti modelli di rete semplicemente modificando dei parametri generali.
Abbiamo quindi reso possibile la creazione di istanze di reti *fully connected* di cui poter specificare un numero variabile di *hidden layer* e neuroni per livello. Ogni *layer* può utilizzare una funzione di attivazione differente tra quelle implementate: lineare, sigmoidea, tangente iperbolica e ReLU. Il modello può utilizzare due differenti *loss function: mean squared error* e *mean euclidean error*.

2.2 Preprocessing

Al fine di ottimizzare le performance per il task di classificazione nei tre dataset MONK, è stato necessario pre-elaborare i dati secondo l'*enconding one-hot*. Questa codifica, adatta per attributi di tipo categorico, serve a esplicitare l'informazione contenuta nella assegnazione dell'attributo. Ogni variabile categorica, infatti, viene esplicitata in n differenti attributi, pari al numero di classi che l'attributo può assumere. Nello specifico, per il dataset MONK, si è passati da un numero di 7 attributi a 18. Questo approccio ha nettamente migliorato sia i tempi che le performance di convergenza sul task.

2.3 Inizializzazione dei pesi

Per evitare una inizializzazione squilibrata dei pesi, che avrebbe potuto causare la saturazione delle unità, abbiamo implementato la possibilità di inizializzare i pesi secondo le strategie proposte prima da Glorot-Bengio e più recentemente rivisitate da He-Zhang-Ren-Su.
Ogni peso, in fase di inizializzazione, assume un valore compreso tra:

$$var(W) = \pm \frac{2}{neuron_{\text{in}}} \tag{1}$$

dove $\text{neuron}_{\text{in}}$ sta per il numero di neuroni nel *layer* precedente che propagano il segnale al *layer* in analisi.

2.4 Validazione

Al fine di ottimizzare l'utilizzo dei dati si è optato per una K-Fold Cross Validation. L'utilizzo di k elevati, da una parte comporta un maggiore sforzo computazionale, ma dall'altra permette di ottenere un modello più robusto, che riesce a gestire in maniera migliore la variazione tra dati in fase di allenamento e quelli proposti in fase di test.
Ciò nonostante, sui task proposti dal MONK Dataset abbiamo optato per una semplice suddivisione del dataset di training in due parti (Training Set del 70% e Validation Set del restante 30%) senza Cross Validation, in quanto è stato utilizzato come semplice *benchmark* per testare la corretta implementazione della rete.

Per il dataset MLCUP, invece, si è scelto un valore k = 7, in modo da ottenere un training set di ampiezza comunque considerevole e non rallentare troppo i tempi di computazione. Il metodo per la suddivisione del dataset di training richiede, quindi, un parametro k per il quale dividerà l'interezza del training set. Se il training non è divisibile perfettamente per k, la parte corrispondente al resto della divisione verrà assegnata automaticamente all'ultima partizione.

2.5 Grid Search

Per individuare la topologia di network più efficace, abbiamo implementato la Grid Search, testando le performance della rete con varie combinazioni di parametri. Inizialmente abbiamo impostato una Grid Search generale, per poterci orientare sull'insieme di modelli sui quali concentrare la nostra ricerca. Per ogni combinazione testata abbiamo allenato il modello per 150 epoche e valutato le sue performance. Il numero di unità input è stato lasciato fisso a 10, e quello di unità output a 2, mentre per ognuno dei parametri qui di seguito elencati, abbiamo valutato ognuno dei rispettivi valori:

1. numero di *hidden layer*: 1, 2, 3, 4
2. numero di neuroni per *layer*: 6, 10, 20, 40
3. funzione di attivazione (per *layer*): tanh, ReLU e lineare (per output *layer*)
4. eta (*learning rate*): 0.01, 0.001, 0.0001
5. lambda: 0.0001, 0.001

Una volta ottenuti i modelli più interessanti, si è passati ad una Grid Search più raffinata, indirizzata all'individuazione degli iper-parametri più performanti in seguito ad un training di 500 epoche per combinazione. I parametri usati sono stati i seguenti, testati su strutture con 1 o 2 *hidden layer* con 10 o 20 neuroni per *layer* (valori che erano risultati migliori nella fase precedente):

1. eta (learning rate): 0.002, 0.006, 0.008, 0.02, 0.06, 0.08
2. alpha (momentum): 0.2, 0.6, 0.8
3. lambda (L2): 0.0001, 0.001
4. *activation function*: tanh e ReLU e lineare (per output *layer*

Considerando anche i diversi valori di numero di *layer* e unità per *layer*, il totale di combinazioni testate è stato pari a 720 (18 combinazioni di parametri per 16 architetture differenti nella prima *grid search*, 108 combinazioni per 4 architetture nella seconda).

3 Esperimenti

3.1 MONK Dataset

Il MONK Dataset risulta particolarmente sensibile all'inizializzazione dei pesi, per via dei pochi dati disponibili. Tuttavia, con un *enconding one-hot* il task non risulta particolarmente complesso. Difatti, nella maggior parte dei test, si raggiungono risultati ottimali con *accuracy* del 100% sia su training che su test set per MONK1 e MONK2 impiegando solamente 4 unità (3 nel *layer* in ingresso e 1 in output). Il task proposto per MONK3 risulta invece più complesso: non si raggiunge una precisione ottimale come nei due precedenti casi ma, si attesta una *accuracy* media del 95%. I valori ottenuti sono, comunque, in accordo con quelli presentati in *The MONK's Problems - A Performance Comparison of Different Learning algorithms*.

Task	Hyperparameters	MSE	Accuracy
MONK1	4 unità eta: 0.3 alpha: 0.6	TR = 0.00107 TS = 0.00123	TR = 100% TS = 100%
MONK2	4 unità eta: 0.3 alpha: 0.6	TR = 0.00041 TS = 0.00051	TR = 100% TS = 100%
MONK3	4 unità eta: 0.03 alpha: 0.6	TR = 0.01694 TS = 0.02360	TR = 98% TS = 95%

Tabella 1: Risultati MONK Dataset.

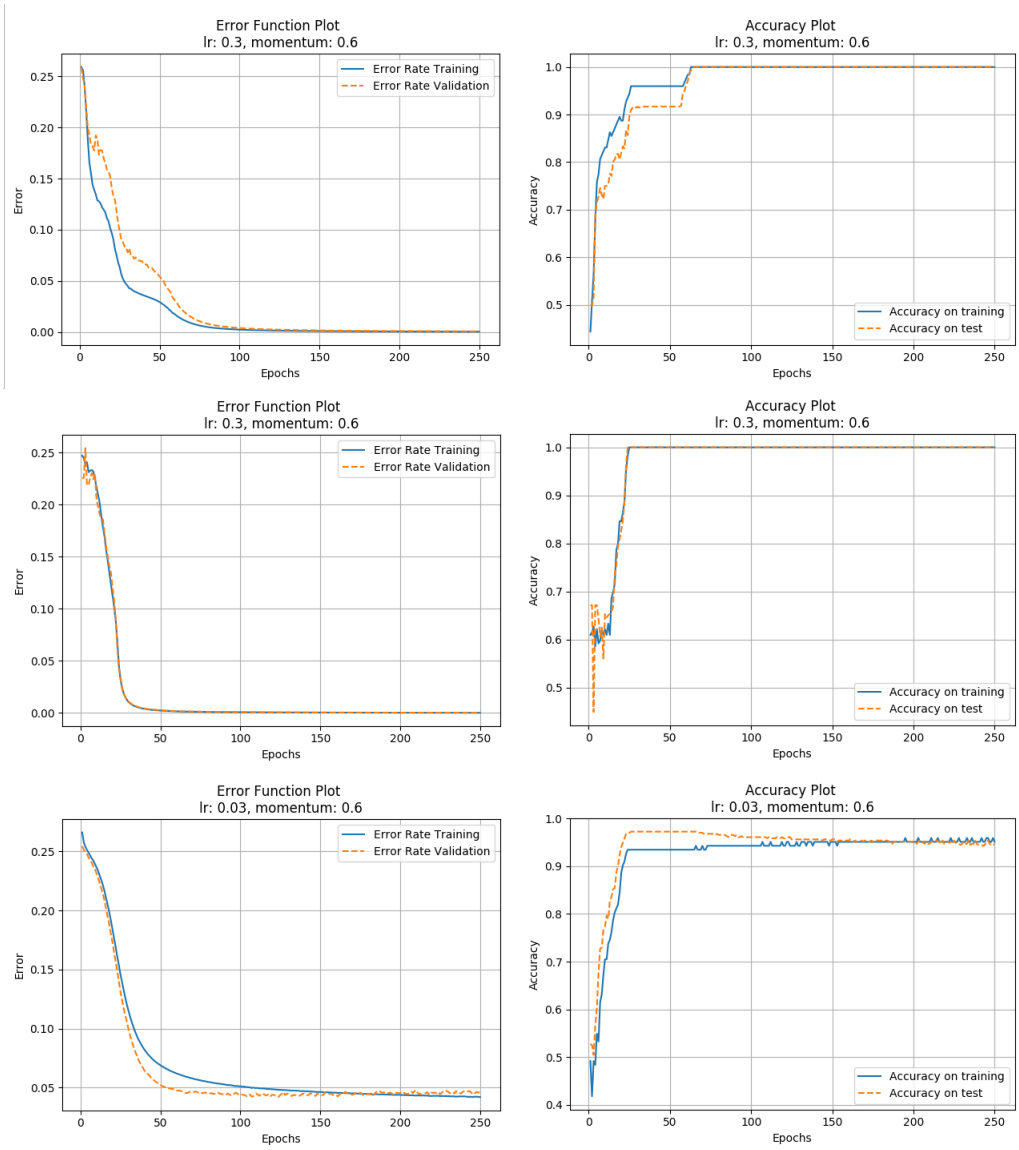

Figura 1: Plot MONK Tasks: Error (Mean Squared Error) and Accuracy

3.2 MLCUP Dataset

Per quanto riguarda il dataset MLCUP, per riuscire a scegliere i parametri da poter usare nel modello definitivo, ci siamo mossi in più passi. Innanzitutto, abbiamo effettuato una *grid search* generale, utilizzando i parametri riportati nella sezione 2.5, ed ottenendo come risultato i valori di errore sul training set e sul validation set esposti qui di seguito.

n hid.layer	6 nn x layer	10 nn x layer	20 nn x layer	40 nn x layer
1	TR = 1.49318 TS = 1.46416	TR = 1.27841 TS = 1.31664	TR = 1.26291 TS = 1.35442	TR = 1.27773 TS = 1.3951
2	TR = 1.39079 TS = 1.48617	TR = 1.28629 TS = 1.35171	TR = 1.27838 TS = 1.35754	TR = 1.33952 TS = 1.37364
3	TR = 1.47518 TS = 1.51242	TR = 1.35011 TS = 1.41647	TR = 1.34318 TS = 1.42094	TR = 1.41263 TS = 1.40744
4	TR = 1.57386 TS = 1.7515	TR = 1.41472 TS = 1.8356	TR = 1.42208 TS = 1.48976	TR = 1.4707 TS = 1.58569

Tabella 2: Risultati prima *grid search* MLCUP18 Dataset.

Dai dati ottenuti abbiamo identificato le strutture più interessanti confrontando gli errori e gli andamenti dell'errore in fase di training e validazione. La scelta è ricaduta sulla "famiglia" di modelli con uno o due *hidden layer* e per ognuno di questi 10 o 20 neuroni. Abbiamo quindi testato queste due categorie per determinare il modello vincente su cui poi rifinire la scelta degli iper-parametri. I risultati ottenuti tramite Grid Search con K-Fold Validation (con k pari a 7) sono riportati nella tabella seguente:

n hid. layer	10 neurons x layer	20 neurons x layer
1	TR = 1.24991 TS = 1.34594	TR = 1.22491 TS = 1.30473
2	TR = 1.27213 TS = 1.3543	TR = 1.28751 TS = 1.3602

Tabella 3: Risultati seconda *grid search* MLCUP18 Dataset. Eta: 0.001

Visti i risultati, abbiamo infine effettuato una ricerca per trovare la migliorore combinazioni di iper-parametri, focalizzandoci su una rete con 1 *hidden layer* da 20 neuroni ed eta di basso valore, in modo da favorire una discesa

del gradiente più lineare. Inoltre, si è scelto il parametro di regolarizzazione lambda più basso fra quelli testati poiché abbiamo ritenuto che fosse già sufficiente a prevenire l'*overfitting* dei dati.

eta	alpha: 0.2	alpha: 0.6
0.05	TR = 1.30116 TS = 1.33471	TR = 1.35364 TS = 1.37825
0.01	TR = 1.27678 TS = 1.30015	TR = 1.31766 TS = 1.36015
0.001	TR = 1.31438 TS = 1.36648	TR = 1.42256 TS = 1.36601
0.005	TR = 1.28468 TS = 1.32772	TR = 1.41353 TS = 1.4883
0.007	TR = 1.30552 TS = 1.31866	TR = 1.44824 TS = 1.50966
0.009	TR = 1.317 TS = 1.331	TR = 1.4751 TS = 1.36769

Tabella 4: Risultati terza *grid search* MLCUP18 Dataset.

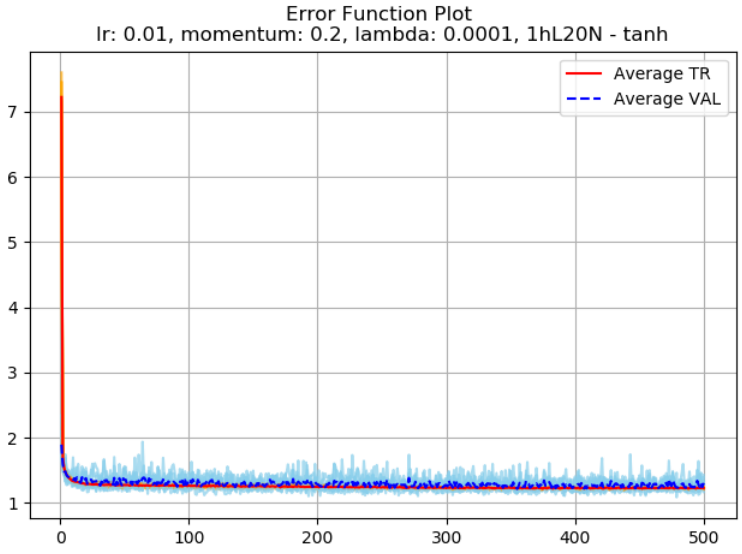

Figura 2: modello finale, 10-fold validation, eta: 0.01, alpha: 0.2 - con MSE

Prima di far girare il modello sul test set per la competizione, abbiamo nuovamente allenato il modello sull'interezza del training set (70% del dataset fornito) e testato sul restante 30% di dataset tenuto separato durante la fase di *model selection*.

# hidden layer	1
# neuroni per layer	20
eta	0.01
alpha	0.2
lambda	0.0001

Tabella 5: Scelta finale dei parametri per MLCUP18 Dataset.

Errori finali: TR: 1.27678 | TS: 1.30015

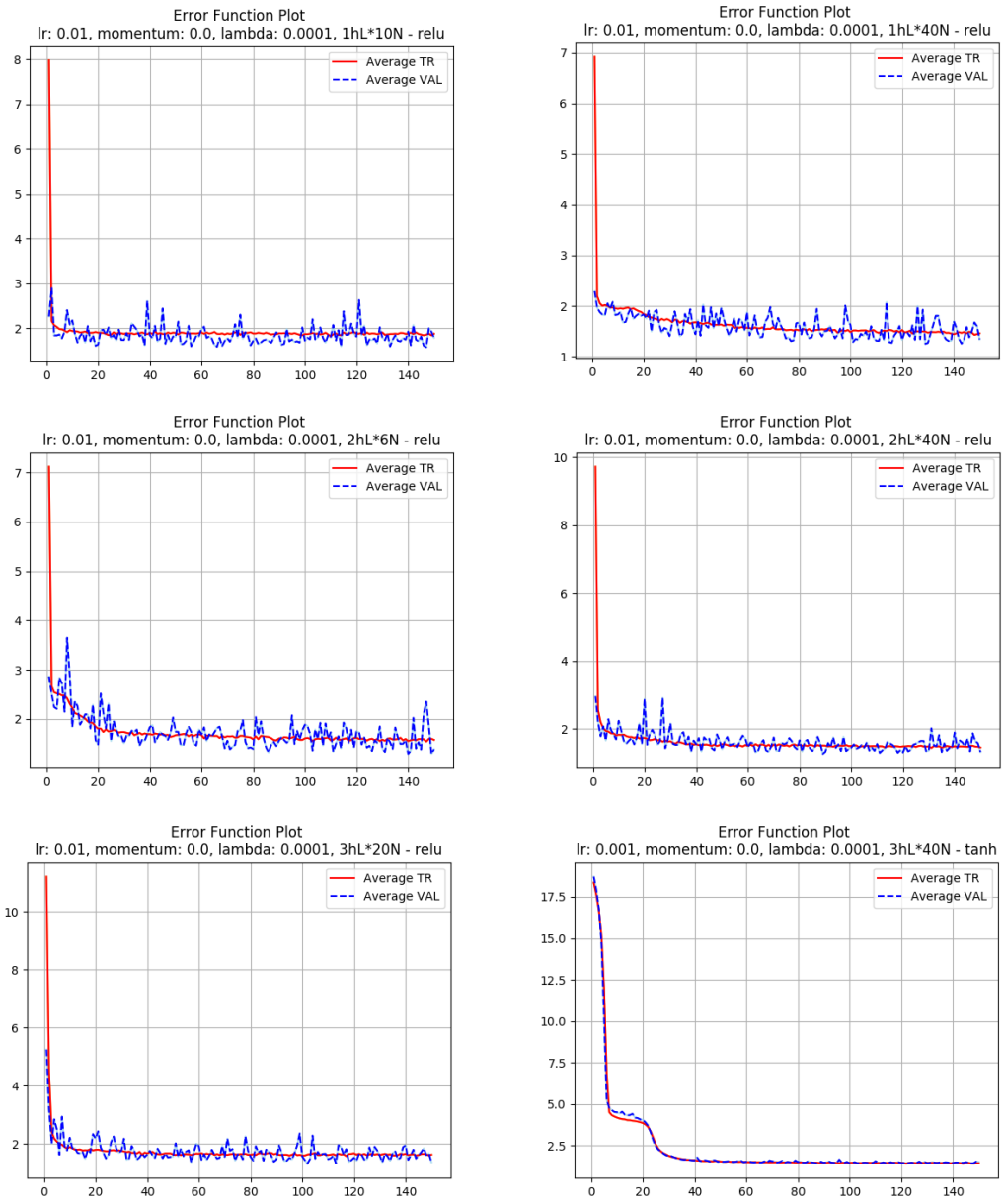

Figura 3: Plot ottenuti tramite *grid search* con MSE

4 Conclusioni

Il progetto è stato un'ottima occasione per toccare con mano una parte degli argomenti trattati durante il corso. Abbiamo potuto constatare come e in che misura gli iper-parametri influenzino i tempi e la stabilità di convergenza di un modello. La scelta del progetto A, sebbene ci abbia posti davanti ad alcuni problemi implementativi come quello dell'efficienza di computazione, ci ha permesso di comprendere a fondo diversi concetti teorici e il funzionamento di una semplice rete neurale artificiale, riuscendo a demistificare quello che prima risultava piuttosto oscuro.

5 Bibliografia

1. http://mars.gmu.edu/handle/1920/1685
2. https://arxiv.org/pdf/1502.01852.pdf