UNIVERSITÀ DI PISA

Laurea Magistrale in Data Science and Business Informatics

MACHINE LEARNING

Progetto A

A.A. 2018/2019

UNIVERSITÀ DI PISA

Fabrizio Massidda fabrizio.massidda3@gmail.com (ML - 654AA)

Daniele Atzeni daniele.atze@gmail.com (ML - 654AA)

25/03/2019

Sommario

1 INTRODUZIONE

In questo report verranno mostrati i risultati sui dataset MONK1, MONK2, MONK3 e sul dataset ML-CUP2018 della Rete Neurale da noi sviluppata, allenata con gli algoritmi Stochastic Gradient Descent (SGD) e ADAM Optimization. Tali risultati saranno accompagnati da considerazioni riguardo le differenti performance osservate al variare degli iperparametri e dell'algoritmo di apprendimento.

2 METODO

2.1 Librerie utilizzate

La Rete Neurale è stata sviluppata utilizzando la versione 3 del linguaggio Python. Il paradigma di programmazione utilizzato è object-oriented.

Sono state utilizzate le librerie NumPy, per la manipolazione di matrici e la creazione di numeri pseudo-casuali, Matplotlib per la produzione di grafici, Math per il calcolo scientifico, e Copy per motivi tecnici.

2.2 Scelte implementative

Le classi e i metodi implementati permettono di istanziare una rete con un numero di hidden layer e di unità per layer a scelta. La topologia della rete è fissa ed è quella classica, cioè ogni unità di un layer è connessa a tutte le unità del layer successivo. Per ogni hidden layer è possibile selezionare una funzione di attivazione a scelta tra sigmoid, tangente iperbolica, identità, ReLu. Si possono inoltre selezionare i seguenti iperparametri:

- Learning rate, disponibile in tre tipologie: costante, con step decay e con exponential decay;
- Alpha, cioè il parametro del momentum;
- Minibatch size;
- Numero massimo di epoche;
- Lambda, cioè il parametro della regolarizzazione L2;
- Classification, un parametro booleano che stabilisce la funzione di attivazione nell'ouput layer: sigmoid per task di classificazione, identità per task di regressione;
- Tipologia dell'algoritmo, uno tra SGD e ADAM.

Per quanto riguarda l'algoritmo ADAM, gli iperparametri aggiuntivi, denominati comunemente con beta1, beta2 ed epsilon, sono fissi e hanno un valore di default pari a 0.9, 0.999 e 10^(-8) rispettivamente.

2.3 Preprocessing

Per i tre dataset MONK, è stato necessario preprocessare i dati. I dataset nel formato originale presentano una colonna target corrispondente alla classe da predire, e sei attributi. La colonna target assume valore binario zero o uno, mentre gli attributi possono assumere valori da 1 a 4. La fase di preproccessing dei dati utilizza un encoding 1-of-k, cioè si sostituisce ogni attributo con un numero di attributi binari pari al numero di valori distinti assunti dall'attributo stesso. Nello specifico se un attributo A assume valori 1 e 2, la codifica sarà effettuata utilizzando due attributi A1, A2, che assumono valori 1 o 0 a seconda del valore di A.

Per il dataset ML-CUP2018 sono stati analizzati i risultati sia senza un preprocessing dei dati, sia applicando normalizzazione Z-Score e Min-Max, senza aver riscontrato cambiamenti rilevanti, perciò si è optato per non preprocessare i dati.

2.4 Validazione

Per la fase di validazione, è stata usata la k-fold cross validation, in cui il parametro k è stato fissato a 5 per i MONK dataset e a 10 per il dataset ML-CUP2018 durante la grid search. Il valore medio dell'errore e la deviazione standard dei risultati della Cross Validation sono stati comparati per selezionare le migliori combinazioni di iperparametri con cui riallenare il modello sull'intero Training set. La grid-search per i dataset MONK non è stata molto approfondita, dal momento che i risultati si sono rivelati più che soddisfacenti con un gran numero di combinazioni degli iperparametri. La funzione di attivazione scelta è tanh, l'algoritmo è SGD ed i range selezionati per questa grid_search sono stati:

- η pari a uno tra 0.01, 0.03, 0.07, 0.1;
- α da 0.7 a 0.9 con uno step di 0.1;
- λ pari a uno tra 0.01, 0.1, 0.2, soltanto nel MONK3;
- numero di layer tra 1 e 3;
- numero di hidden units con valori da 5 e 20 con step pari a 5;
- minibatch size appartenente alle potenze di due tra 16 e 64, a cui va aggiunta la prova con la versione batch dell'algoritmo.

3 ESPERIMENTI

3.1 Risultati MONK

I risultati ottenuti sia su training set che test set dei dataset MONK sono sempre abbastanza buoni (difficilmente si scende sotto il 90% di accuracy) e, come visibile in Tabella 1 si raggiunge un'accuracy del 100% su MONK 1 e MONK 2. Il MONK 3, essendo stato creato utilizzando un noise pari al 5%, risulta essere il dataset più "problematico". Onde evitare overfitting sul training set dovuto alla presenza del rumore, per questo dataset è stato aggiunto anche il parametro di regolarizzazione, cosa non avvenuta per gli altri due dataset in

cui non è possibile andare in overfitting. La sensibilità dei risultati all'inizializzazione dei pesi è stata resa ininfluente utilizzando 10 diverse inizializzazioni durante ogni allenamento della rete e selezionando solamente i pesi più performanti.

DATASET	IPERPARAMETRI	MSE	ACCURACY
MONK 1	1 hidden layer 15 unità $\lambda = 0$ $\alpha = 0.8$ $\eta = 0.1$ minibatch = 32	TR = 0.0026 TS = 0.0095	TR = 100 % TS = 100 %
MONK 2	1 hidden layer 10 unità $\lambda = 0$ $\alpha = 0.8$ $\eta = 0.1$ minibatch = 32	TR = 0.0003 TS = 0.0009	TR = 100 % TS = 100 %
MONK 3	3 hidden layer 10 unità $\lambda = 0.2$ $\alpha = 0.8$ $\eta = 0.1$ minibatch = 64	TR = 0.0386 TS = 0.0525	TR = 97 % TS = 95 %

Tabella 1 - Risultati per i tre dataset MONK.

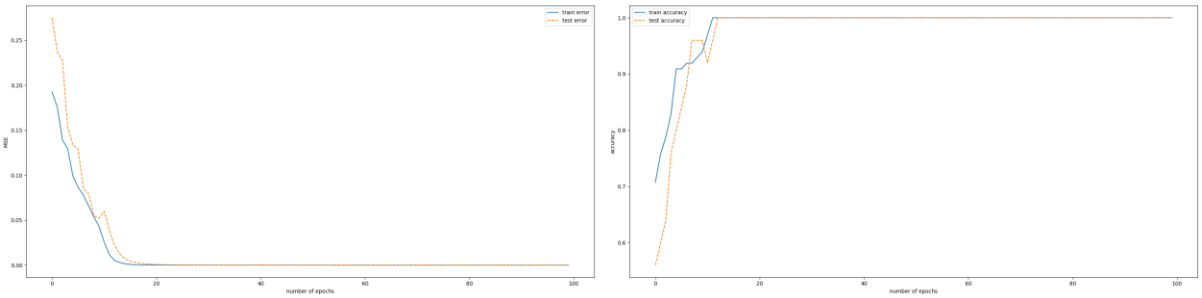

Figura 1 - MSE e accuracy su MONK 1.

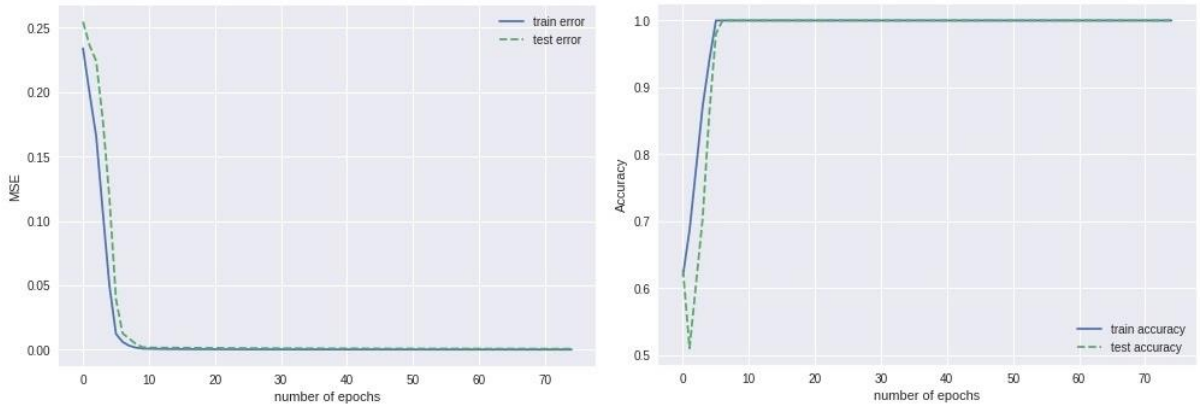

Figura 2 - MSE e accuracy su MONK 2.

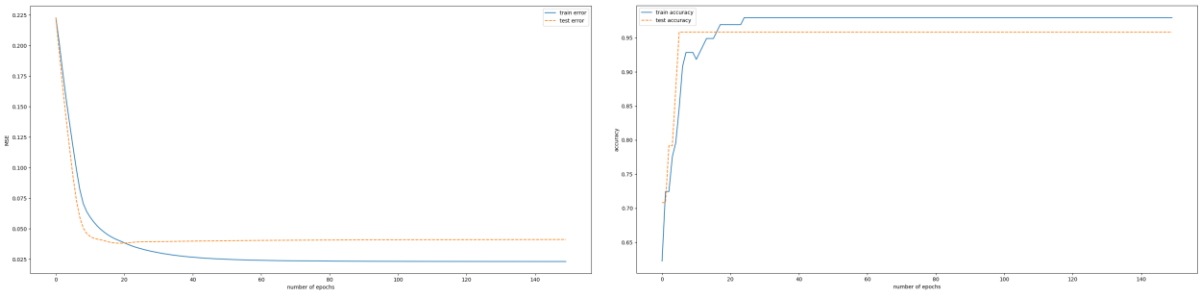

Figura 3 - MSE e accuracy su MONK 3.

3.2 Risultati CUP

Per allenare e testare la rete utilizzando il dataset ML-CUP2018, si è diviso tale dataset in una parte di training (75%), sulla quale è stata effettuata una 10-Fold CV, e una parte di test (25%).

3.2.1 Inizializzazione dei pesi

Per l'inizializzazione dei pesi si è optato per la Xavier initialization, per la quale ogni elemento della matrice dei pesi assume un valore compreso nel range $[-2\ (units_{in}+units_{out}); +2(units_{in}+units_{out})]$. Tale scelta sembra garantire una maggiore indipendenza dei risultati finali rispetto all'inizializzazione dei pesi.

3.2.2 Grid search

Per la scelta degli iperparametri è stata effettuata una grid search per osservare le diverse performance della rete. La funzione di attivazione utilizzata è la tangente iperbolica in quanto è risultata più semplice la fase di allenamento della rete. I range utilizzati per l'analisi sono i seguenti, differenti a seconda della scelta dell'algoritmo tra SGD e ADAM:

- η compreso tra le potenze di 10 da 10^(-5) e 10^(-3), inclusi i valori intermedi tra una potenza e l'altra per l'algoritmo SGD, tra le potenze di 10 da 10^(-3) a 10^(-1) inclusi i valori intermedi tra le potenze per l'algoritmo ADAM;
- α da 0.7 a 0.9 con uno step di 0.1 per l'algoritmo SGD, pari a 0 per l'algoritmo ADAM;
- λ compreso tra 0.001, 0.1, con step di 0.005 ;
- numero di layer tra 1 e 4;
- numero di hidden units con valori da 10 e 100 con step pari a 5;
- minibatch size appartente alle potenze di due tra 32 e 256, a cui va aggiunta la prova con la versione batch dell'algoritmo per l'algoritmo SGD, appartenente alle potenze di due da 16 a 128 per l'algoritmo ADAM.

Dopo una preliminare grid search con i suddetti valori, si è ripetuta l'operazione, andando ad escludere dalla ricerca quei valori che portavano a un andamento eccessivamente instabile della learning curve, anche dopo aver utilizzato un learning rate descrescente, o che comportavano un errore euclideo medio e deviazione standard eccessivamente superiore alla media delle altre combinazioni di iperparametri.

UNITS, # LAYER, λ	IPERPARAMETRI (SGD/ADAM)	MEE - STD
50 unità 2 hidden layers λ = 0.1	η = 0.00005, α = 0.8, mb = 128	MEE = 1.1615 STD = 0.0964
	η = 0.001, mb = 16	MEE = 1.1787 STD = 0.1017
15 unità 3 hidden layers λ = 0.2	η = 0.00001, α = 0.9, mb = 128	MEE = 1.2094 STD = 0.1158
	η = 0.002, mb = 16	MEE = 1.1862 STD = 0.0985
10 unità 4 hidden layers λ = 0.2	η = 0.00001, α = 0.9, mb = 256	MEE = 1.1544 STD = 0.1429
	η = 0.003, mb = 16	MEE = 1.1938 STD = 0.1506
15 unità 4 hidden layers λ = 0.01	η = 0.00001, α = 0.9, mb = 128	MEE = 1.1616 STD = 0.0997
	η = 0.0015, mb = 8	MEE = 1.2213 STD = 0.1463
40 unità 4 hidden layers λ = 0.01	η = 0.00001, α = 0.9, mb = 128	MEE = 1.1964 STD = 0.1258
	η = 0.0005, mb = 16	MEE = 1.1299 STD = 0.1769

Tabella 2 - MEE e STD risultanti dalla 10-fold cv al variare degli iperparametri.

3.2.3 Modello finale

Per il modello finale si è deciso di utilizzare i seguenti iperparametri, sia per i risultati ottenuti sia per ragioni di stabilità.

IPERPARAMETRI	TRAINING + VL (75%)	TEST (25%)
50 unità, 2 hidden layers, $\lambda = 0.1$, $\eta = 0.00005$, $\alpha = 0.8$, mb = 128, SGD	0.9228	1.1415

Tabella 3 - Risultati modello finale.

Il modello è stato riallenato sull'intero TR (75% dataset) per poter poi effettuare la model assessment testando sul 25% del dataset a disposizione.

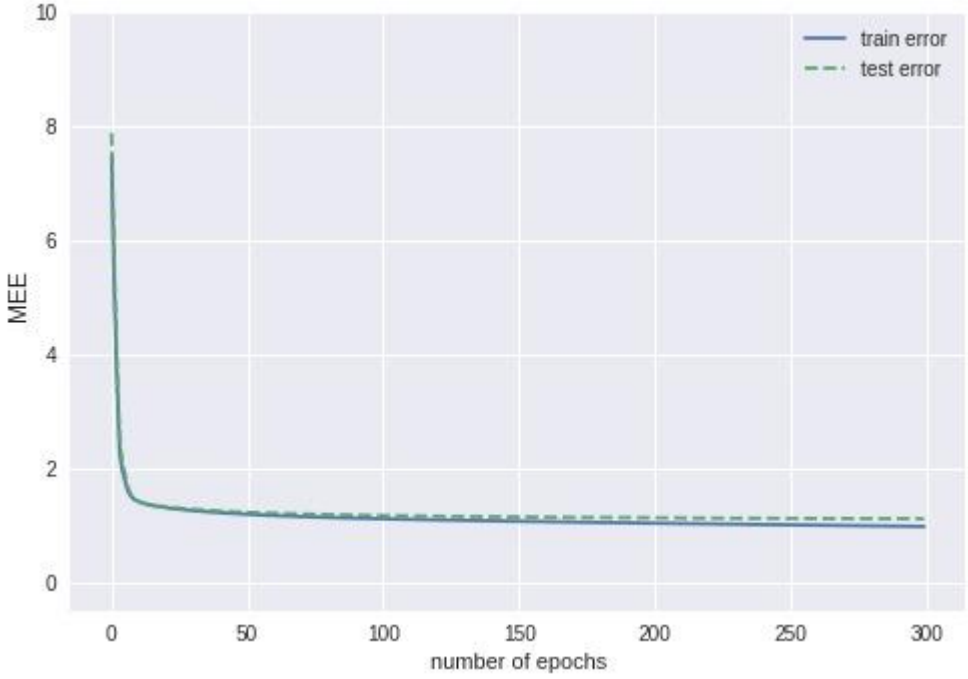

Figura 4 - MEE del modello finale su TR e TS

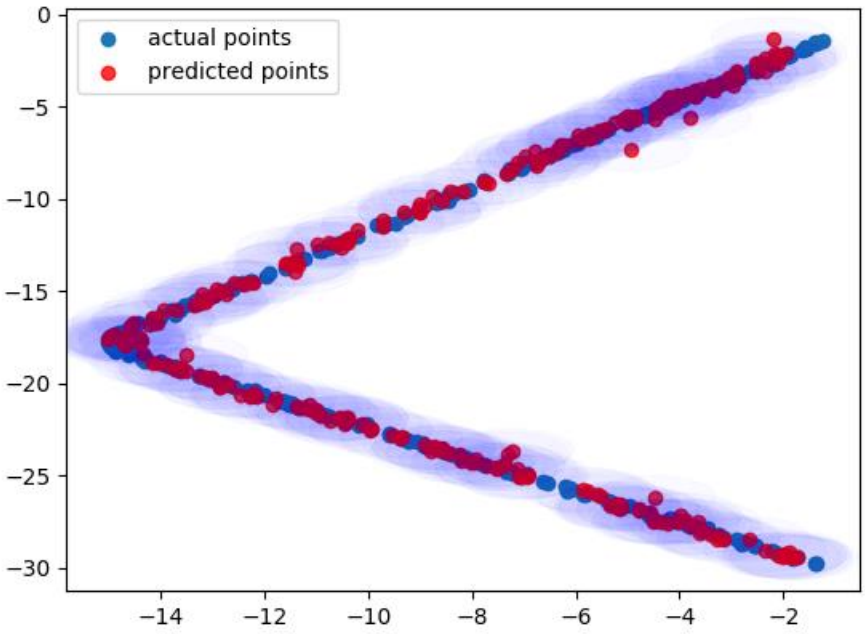

Figura 5 - Previsioni sul TS (25%)

4 CONCLUSIONI

Per concludere, i dati mostrati indicano come le differenti combinazioni di iperparametri influiscano sul processo di learning, sia in termini di prestazioni che di stabilità.

L'algoritmo ADAM si è dimostrato più performante anche con valori molto bassi di minibatch e più elevati di learning rate. Ciò rende l'algoritmo molto più rapido nella fase di allenamento (si possono aggiornare i pesi molte volte in una sola epoca), nonostante i risultati finali si siano rivelati simili rispetto all'algoritmo SGD.

In generale, grazie allo sviluppo del progetto, ci è stato possibile toccare con mano e veder realizzate in termini pratici parte delle tecniche utilizzate nel Machine Learning, che sarebbero altrimenti rimaste nozioni teoriche. Inoltre, la scelta del progetto di tipo A ci ha posto davanti alcune sfide di carattere implementativo, presentandosi perciò come una buona occasione per approfondire la nostra conoscenza della programmazione in Python e dell'utilizzo delle librerie fondamentali, soprattutto NumPy.

Per la visione dei dataset e dei codici si rimanda al seguente link: https://github.com/daniele-atzeni/Machine-Learning-project.

Acconsentiamo alla pubblicazione dei nostri nomi e dei risultati per il ranking preliminare e finale.