Progetto Machine Learning

Leonardo Frioli / Luigi Quarantiello
l.frioli@studenti.unipi.it
l.quarantiello@studenti.unipi.it

ML (654AA), 2019/2020
Date: *10/01/2020*
Type of project: **A**

ABSTRACT

Il modello utilizzato è una rete neurale di tipo MLP che implementa un algoritmo di backpropagation standard e permette l'utilizzo di momentum e regularization.
Per la cup, è stata impiegata una rete con due hidden layers, ognuno composto da 16 unità: ogni unità è connessa a tutte quelle del livello precedente e a tutte quelle del successivo.
La validation è stata eseguita dividendo i dati con la tecnica hold-out ed eseguendo una grid search.

1. INTRODUZIONE

Il progetto richiede di implementare un MLP per problemi di classificazione e regressione. L'obiettivo del report è di mostrare le conoscenze acquisite nella gestione degli hyperparameters, come la capacità di selezionarli e sintonizzarli (tuning). Mostreremo, inoltre, come i parametri vanno ad influenzare la capacità di apprendimento del modello. L'idea generale è di acquisire sufficiente conoscenza a priori sull'uso degli hyperparameters in modo che, dato un parametro e alcuni suoi valori, si riesca a sapere in precedenza l'influenza che esso può avere sul modello.

Il modello utilizzato è una rete neurale di tipo multilayer perceptron che implementa un algoritmo di backpropagation standard e consente l'utilizzo di momentum e regularization.

Si assume che i modelli richiesti per la risoluzione dei due problemi (monks e cup) siano composti da un numero basso di hidden layers.

2. METODO

Essendo il progetto di tipo A, le uniche librerie utilizzate sono **numpy** per la gestione delle matrici dei pesi nella backpropagation e **matplotlib** per i grafici.

La rete è composta da una lista di layers, ognuno dei quali tiene una lista di neuroni. Ogni neurone è collegato a tutti quelli del livello precedente e successivo. Il neurone ha una lista di pesi e ha lo scopo di calcolare l'output attraverso la sua funzione di attivazione. Le funzioni implementate sono due: sigmoid e lineare. Queste funzioni vengono implementate in classi separate così che risulta molto semplice aggiungere nuove funzioni di attivazione.
Per mantenere il codice pulito, ci siamo ispirati a librerie come tensorflow, keras e scikit-learn nella costruzione dell'interfaccia: per esempio il modello viene assemblato da una classe Builder e la classe Dataset permette una gestione uniforme di batch, mini-batch e on-line training.
Per il training abbiamo usato una standard backpropagation (come quella vista in classe); il gradient viene diviso per la dimensione della batch .
Per la regolarizzazione abbiamo utilizzato la Tikhonov regularization; inoltre, il modello consente l'utilizzo sia del momentum classico, sia del Nesterov momentum.

3. ESPERIMENTI

3.1 MONK'S RESULTS

Gli input sono stati preprocessati attraverso il 1-of-k encoding.
La suddivisione del dataset è stata fatta rispettando le direttive presenti nel paper che tratta il monks problem **[1]** usando la tecnica hold-out:
monks-1 e monks-3: 29% TR, 14% VL, 57% TS
monks-2: 39% TR, 19% VL, 42% TS
Le metriche calcolate derivano dalla media delle metriche di 3 modelli allenati con gli stessi parametri. Sono stati usati 3 modelli per questioni di performance della macchina su cui è avvenuto il training. Il MSE del TS è stato calcolato prendendo come output la media degli output dei tre modelli allenati; l'ACC, PREC e RECALL del TS sono state calcolate prendendo come output il valore più frequente dato dai tre modelli (i.e il voto).
Durante la fase di sviluppo sono state eseguite diverse run di training per testare il funzionamento del modello e per avere un'idea dell'influenza dei vari parametri. I parametri usati durante queste prove sono stati presi dal paper **[1]**, per poi essere aggiustati nella computazione dei risultati mostrati in [**Tabella 1**].

Tabella 1. Risultati relativi al MONK's problem

Task	**#Units, eta, lambda, ..**	**MSE (TR/VL/TS)**	**Accuracy (VL/TS) (%)**	**Precision/Recall (TS)(%)**
MONK 1	hidden units: 3 eta: 0.95 alpha: 0.7 weights bound: 0.000009 epochs: 800 batch	mse TR: 0.0254 mse VL: 0.0329 mse TS: 0.0376	acc VL: 100 % acc TS: 99 %	prec: 100% rec: 99%
MONK 2	hidden units: 2	mse TR: 0.0059	acc VL: 100%	prec: 100%

	eta: 0.05 weights bound: 0.000009 epochs: 500 on-line	mse VL: 0.0078 mse TS: 0.011	acc TS: 100%	rec: 100%
MONK3	hidden units: 4 eta: 0.8 alpha: 0.5 weights bound: 0.00009 epochs: 190 batch	mse TR: 0.0729 mse VL: 0.0760 mse TS: 0.074	acc VL: 98% acc TS: 98%	prec: 100% rec: 97%
MONK3 (reg.)	hidden units: 4 eta: 0.95 alpha: 0.7 weights bound: 0.00009 lambda: 0.01 epochs: 390 batch	mse TR: 0.1015 mse VL: 0.1044 mse TS: 0.125	acc VL: 98% acc TS: 96%	prec: 100% rec: 92%

Tabella 2. Grafici relativi al MONK problem

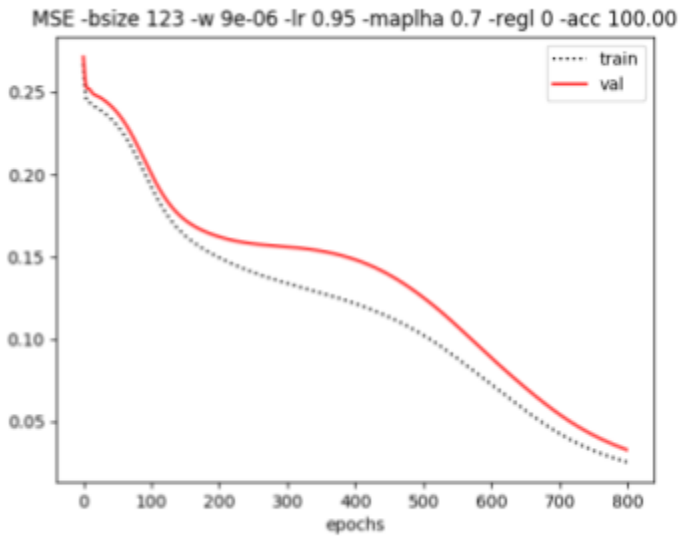

Figura 1. Mean Square Error per il monks-1

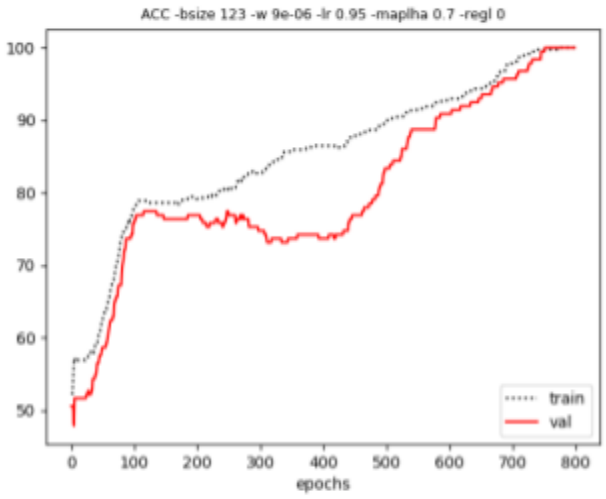

Figura 2. Accuracy per il monks-1

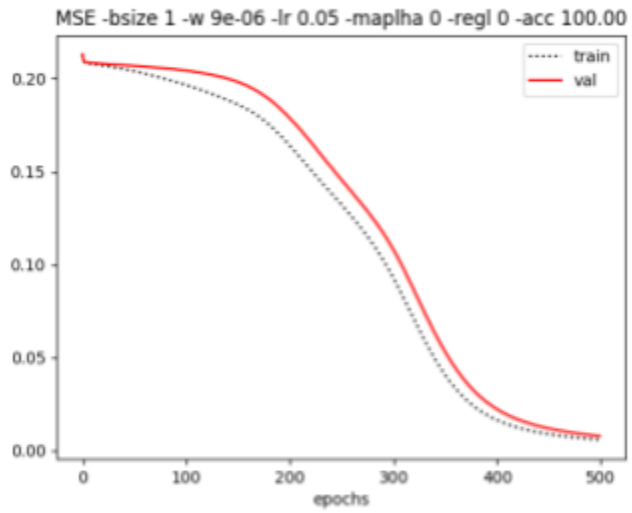

Figura 3. Mean Square Error per ilmonks 2

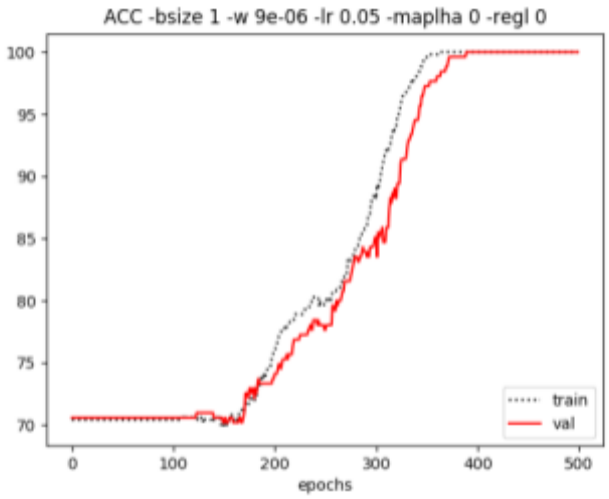

Figura 4. Accuracy per il monks 2

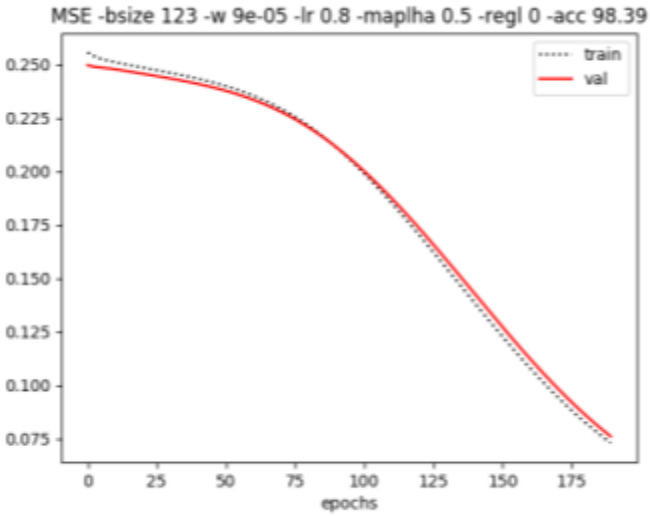

Figura 5. Mean Square Error per il monks 3

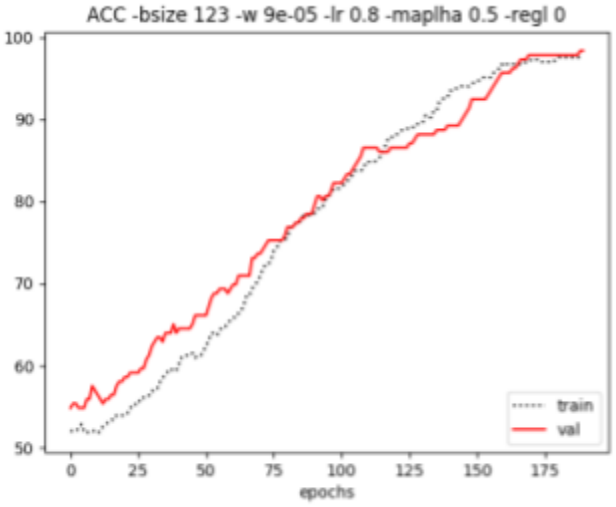

Figura 6. Accuracy per il monks 3

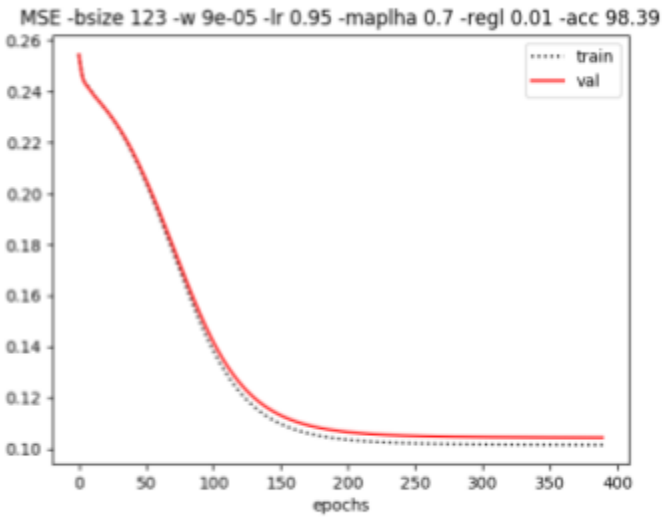

Figura 7. Mean Square Error per il monks 3 + regularization

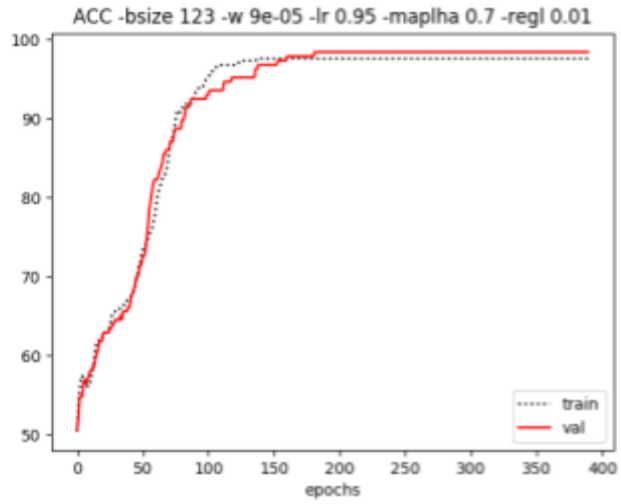

Figura 8. Accuracy per il monks 3 + regularization

3.2 RISULTATI DELLA CUP

Il dataset è stato diviso utilizzando la strategia hold-out. Per la model selection 50% dei dati sono stati usati per il TS, 25% per il VL. Il metodo di training utilizzato è stato maggiormente quello di tipo batch in quanto consente l'utilizzo del momentum e, in generale, produce una learning curve migliore.
Nella fase di model assessment il modello è stato allenato sul 75% dei dati (TS + VL). Per essere indipendenti dall'inizializzazione dei pesi, per ogni run sono stati allenati 3 modelli e i risultati mostrati derivano dalla media dei rispettivi errori. Nella fase di model assessment sono stati sempre usati 3 modelli e l'output usato per il calcolo del MSE e MEE deriva dalla media degli output dei tre modelli.

La grid search è stata eseguita fornendo allo script **cup_main.py** un file json(presente nella directory **model_selection**) con i valori dei parametri sui quali eseguire la ricerca . Lo script genera i plot per MSE, MEE (uno per combinazione) e un file .csv contenente tutte le informazioni relative alle differenti combinazioni provate in modo che possano essere confrontate e ordinate in base ai parametri. Per l'assessment del modello è sufficiente definire il file .json con i parametri scelti e passarlo in input allo script **cup_assessment.py** . Quest ultimo script fornirà i dati relativi all'uso del test set. Lo script per la valutazione del modello finale è tenuto separato in modo che venga eseguito solamente dopo aver scelto i parametri finali per il modello.

La model selection è stata svolta in alcune fasi:

1) Inizialmente sono state eseguite alcune run per capire quali erano i range dei parametri più adatti. Si è poi passati ad un'ampia grid search [**Tabella 3**]. Alla fine

abbiamo sintonizzato i parametri per ridurre l'errore. Questa fase ha portato ha valori di MSE nell'ordine di 5.

2) Abbiamo provato una grid search [**Tabella 5**] normalizzando i dati (min-max rescaling) per vederne gli effetti sull'errore. Il rescaling è stato applicato a tutto il dataset per colonne, usando come valori di min e di max, il minimo e il massimo del training set. Il rescaling, però, non ha portato a valori di errore diversi (in percentuale) rispetto a quelli trovati precedentemente.
3) Sono state eseguite diverse run cercando di sintonizzare i parametri. Le run sono state eseguite inizializzando i parametri con valori prossimi a quelli trovati nella grid search della fase 1. I risultati ottenuti sono stati molto simili a quelli della fase 1 eccetto per un caso mostrato nella [**Tabella 6**]**.**
4) Infine abbiamo provato alcune run cercando di aumentare il numero di nodi per livello, di utilizzare training on-line, minibatch, di usare il nesterov momentum. Il miglior risultato è mostrato nella [**Tabella 7**] .

Tabella 3. Parametri usati nella fase 1 della model selection

epochs	100, 200
training	batch
weights bound	0.000009, 0.00009, 0.009
learning rate	0.1, 0.2, 0.3
momentum alpha	0.5, 0.6, 0.7
regularization lambda	0.01, 0.005, 0.015
hidden layer number	2
hidden layer units	[3,3], [4,4]

Tabella 4. Risultati della grid search nella fase 1 della model selection

epochs: 200, **training:** batch **weights bound:** 0.009, **learning rate:** 0.1 **momentum alpha:** 0.5 **regularization lambda:** 0.015 **hidden layer number:** 2 **hidden layer units:** [3,3]	MSE TR: 25.14 MSE VL: 26.50 [**Figura A.1**] MEE TR: 5.71 MEE VL: 5.88 [**Figura A.2**]

epochs: 200, **training:** batch **weights bound:** 0.009, **learning rate:** 0.1 **momentum alpha:** 0.5 **regularization lambda:** 0.005 **hidden layer number:** 2 **hidden layer units:** [3,3]	MSE TR: 8.33 MSE VL: 8.81 [**Figura A.3**] MEE TR: 3.27 MEE VL: 3.35 [**Figura A.4**]
epochs: 200, **training:** batch **weights bound:** 0.009, **learning rate:** 0.1 **momentum alpha:** 0.7 **regularization lambda:** 0.005 **hidden layer number:** 2 **hidden layer units:** [3,3]	MSE TR: 5.607 MSE VL: 6.083 [**Figura A.5**] MEE TR: 2.675 MEE VL: 2.784 [**Figura A.6**]

La [**Tabella 4**] mostra il processo di "tuning" degli hyperparameters per ridurre l'errore. Aumentando il coefficiente del momentum (alpha) e diminuendo quello della regolarizzazione (lambda), il MSE è stato portato a valori prossimi al 5. La grid search ha mostrato, inoltre, che l'apprendimento non è altamente influenzato dall'inizializzazione dei pesi e che un numero di epoche intorno alle 200 risulta sufficiente.
La grid search è stata eseguita su un processore IntelCore i7-7700HQ e non è stato possibile sfruttare a pieno il parallelismo del processore a causa del linguaggio usato (python3.6). Il tempo di training per le run della fase1 si aggira intorno ai 150 secondi.

Tabella 5. Parametri usati nella fase 2 della model selection

epochs	100, 200, 300
training	batch
weights bound	0.00009
learning rate	0.1, 0.01, 0.05
momentum alpha	0.5, 0.6, 0.7, 0.8
regularization lambda	0.01, 0.005, 0.015
use nesterov	0
hidden layer number	2
hidden layer units	[3,3], [4,4]

Tabella 5. Risultati della grid search nella fase 2 della model selection

epochs: 100, **training:** batch **weights bound:** 0.00009, **learning rate:** 0.1 **momentum alpha:** 0.8 **regularization lambda:** 0.001 **hidden layer number:** 2 **hidden layer units:** [3,3]	MSE TR: 0.139 MSE VL: 0.136 [**Figura A.7**] MEE TR: 0.452 MEE VL: 0.447 [**Figura A.8**]
epochs: 100, **training:** batch **weights bound:** 0.00009, **learning rate:** 0.05 **momentum alpha:** 0.7 **regularization lambda:** 0.001 **hidden layer number:** 2 **hidden layer units:** [3,3]	MSE TR: 0.141 MSE VL: 0.140 [**Figura A.9**] MEE TR: 0.456 MEE VL: 0.453 [**Figura A.10**]

La [**Tabella 5**] mostra i risultati del processo di "tuning" degli hyperparameters per ridurre l'errore. Nel caso del training batch, utilizzando i valori più elevati del coefficiente del momentum (alpha) e quelli più bassi per il coefficiente della regolarizzazione (lambda), si ottiene un MSE di circa 0.14.
Anche in questo caso, abbiamo notato dalla grid search che i risultati non sono fortemente influenzati dall'inizializzazione dei pesi e che è sufficiente un numero di epoche abbastanza basso, intorno a 100.
La grid search è stata eseguita su un processore IntelCore i7-8565U. Il tempo di training per le run della fase 2 si aggira intorno ai 110/120 secondi.

Tabella 6 grid search fase 3

epochs: 200, **training:** batch **weights bound:** 0.009, **learning rate:** 0.1 **momentum alpha:** 0.7 **regularization lambda:** 0.00005 **hidden layer number:** 2 **hidden layer units:** [3,3]	MSE TR: 3.463 MSE VL: 3.886 [**Figura A.11**] MEE TR: 2.086 MEE VL: 2.179 [**Figura A.12**]

Tabella 7 grid search fase 4

epochs: 150, **training:** batch **weights bound:** 0.009, **learning rate:** 0.1 **momentum alpha:** 0.5 **regularization lambda:** 0.00005	MSE TR: 2.194 MSE VL: 2.62 MEE TR: 1.619 MEE VL: 1.721

hidden layer number: 2 **hidden layer units:** [16,16]	

La grid search è stata eseguita su un processore IntelCore i7-7700HQ in 642 secondi a causa dell'elevato numero di nodi e del fatto che vengono allenati 3 modelli per ottenerne la media degli errori.

Il modello selezionato è quello definito dai parametri mostrati nella [**Tabella 7**]. Il modello è stato scelto in quanto presenta i valori più bassi di MSE e MEE trovati durante la ricerca. Per la fase di model assessment e di produzione dei risultati per la CUP ricordiamo che il modello è stato allenato sul 75% dei dati (TS +VL) e gli output sia per il TS sia per la CUP derivano dalla media degli output forniti dai 3 modelli allenati.

Tabella 8. MSE e MEE per il final model

	TR+VL	TS
MSE	2.382	2.371
MEE	1.677	1.716

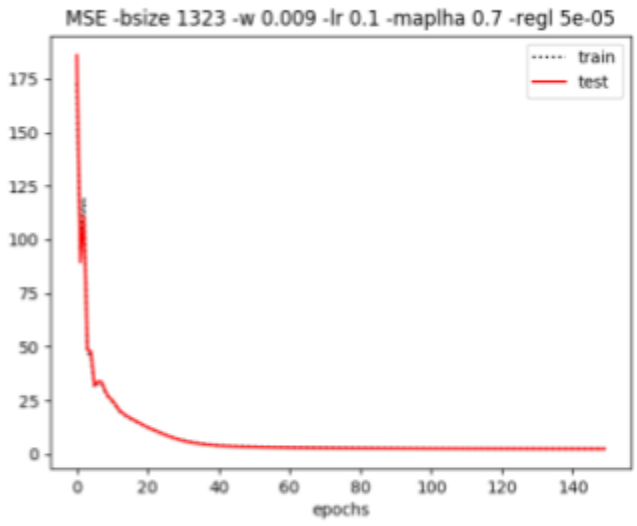

Figura 9. MSE del final model

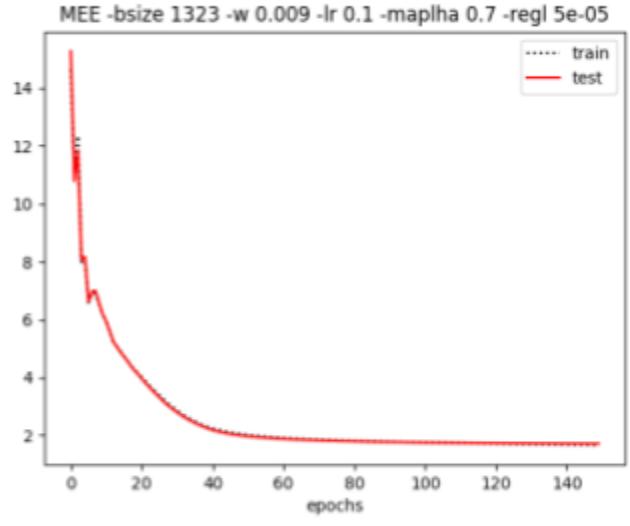

Figura 10. MEE del final model

I vari esperimenti hanno messo in luce l'importanza dei parametri: 'learning rate', 'momentum alpha' e 'regularization lambda', mostrati in particolare nella fase 1 della grid search. Aumentando la learning rate si tende ad ottenere un plot più irregolare senza miglioramenti degli errori. Anche il momentum influenza sulla regolarità del plot, ma ha permesso un abbassamento dell'errore. L'influenza maggiore l'ha avuta il parametro lambda che, se settato a valori opportunamente bassi, consente un miglioramento dell'errore maggiore rispetto a quello apportato dalla variazione degli altri parametri.

4. CONCLUSIONI

Possiamo concludere affermando che il progetto implementato fornisce gli strumenti per la costruzione di un MLP attraverso un'interfaccia modulare e facilmente espandibile. I modelli utilizzati sembrano avere delle buone performance sia nel MONK problem che nella CUP.

Nel MONK problem il raggiungimento di valori elevati di accuracy non ha richiesto una grid search ampia in quanto il modello tende a convergere nella maggior parte dei casi anche con pochissime unità.
La CUP ha richiesto una ricerca più approfondita dei parametri per cercare ogni volta di ridurre di un pò l'errore. Nella CUP si riesce maggiormente a capire e vedere l'importanza dell'inizializzazione dei parametri in quanto è più difficile raggiungere buoni valori di MEE.

Purtroppo siamo stati un pò limitati dalla capacità di computazione delle macchina su cui sono avvenuti gli esperimenti e dall'impossibilità di avere un parallelismo effettivo a causa del linguaggio di programmazione utilizzato.

I risultati del blind test si trovano dentro la directory **report** nel file **poxebur_wikilele_ML-CUP19-TS.csv** .

Acconsentiamo alla pubblicazione dei nostri nomi e dei risultati.

BIBLIOGRAFIA

[1] S.B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. De Jong, S. Dzeroski, S.E. Fahlman, D. Fisher, R. Hamann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, R.S. Michalski, T. Mitchell, P. Pachowicz, Y. Reich H. Vafaie, W. Van de Welde, W. Wenzel, J. Wnek, and J. Zhang: The MONK's Problems A Performance Comparison of Different Learning Algorithms ; Carnegie Mellon University CMU-CS-91-197, December 1991, page 102

APPENDICE. A.

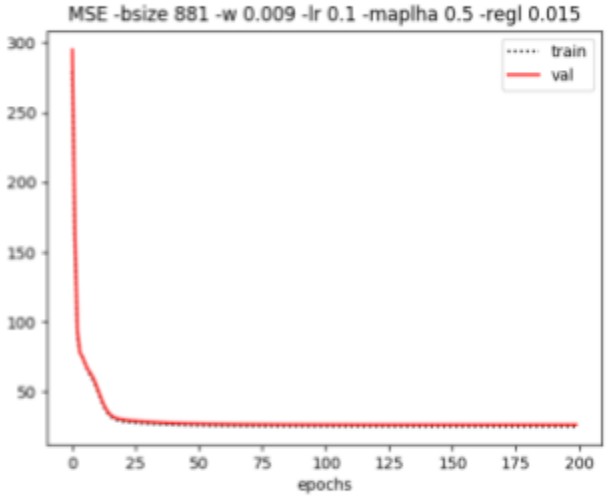

Figura A.1 MSE grid search fase 1

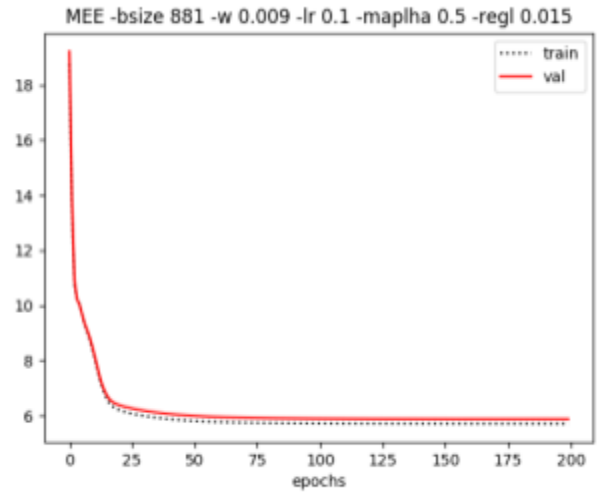

Figura A.2 MEE grid search fase 1

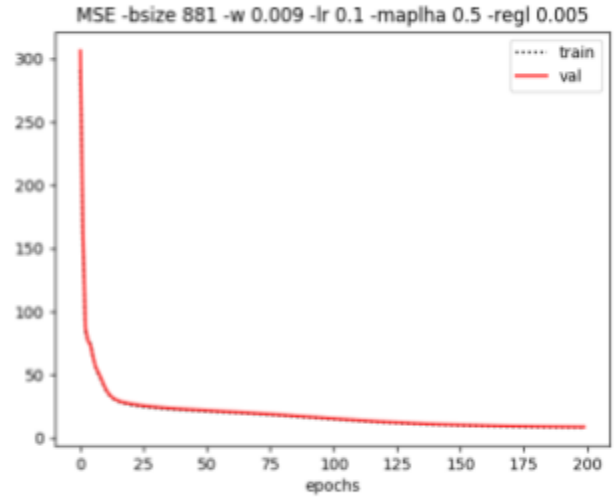

Figura A.3 MSE grid search fase 1

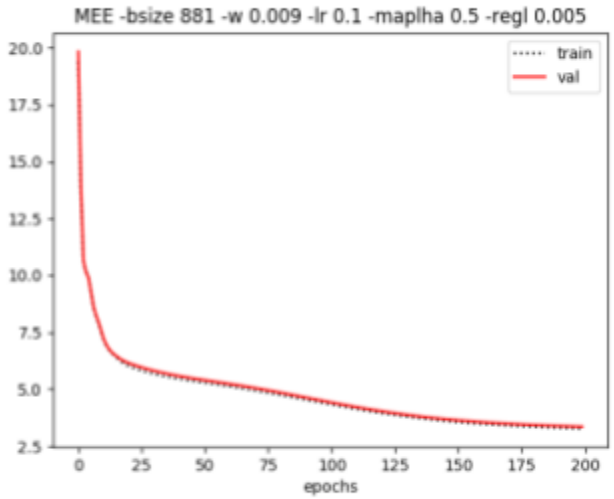

Figura A.4 MEE grid search fase 1

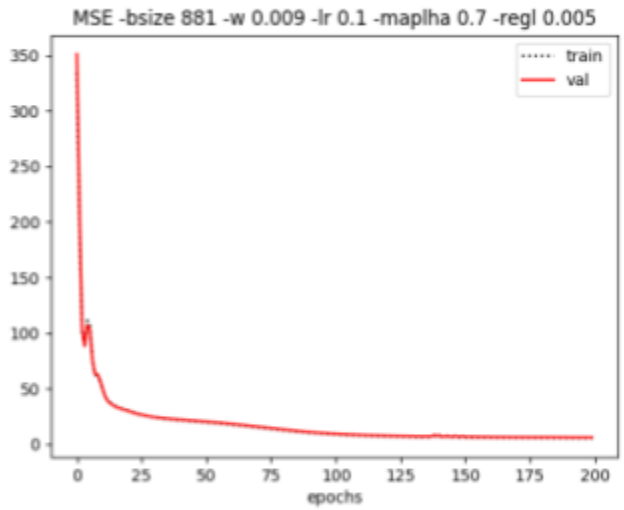

Figura A.5 MSE grid search fase 1

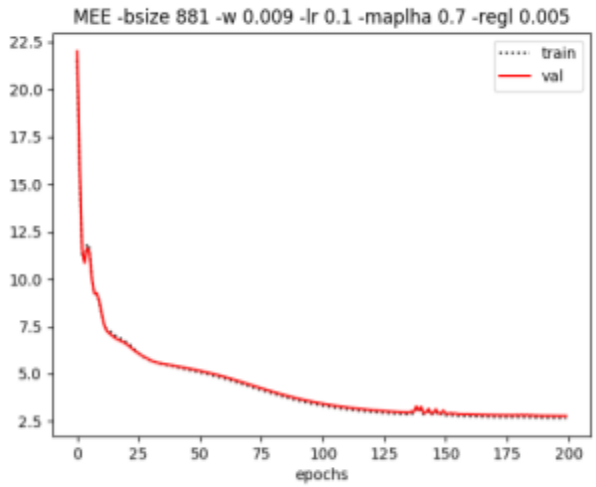

Figura A.6 MEE grid search fase 1

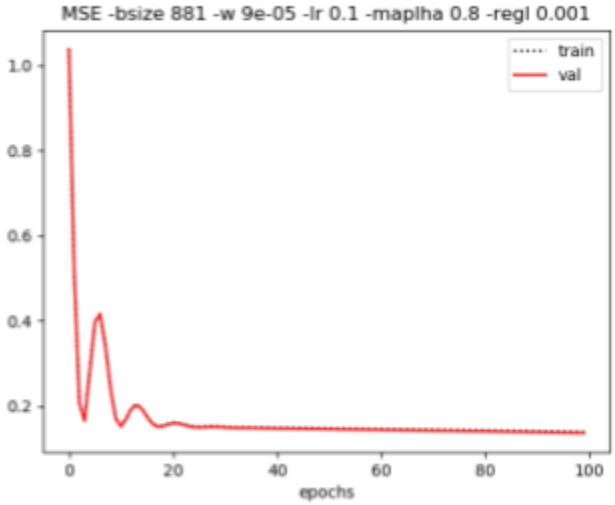

Figura A.7 MSE grid search fase 2

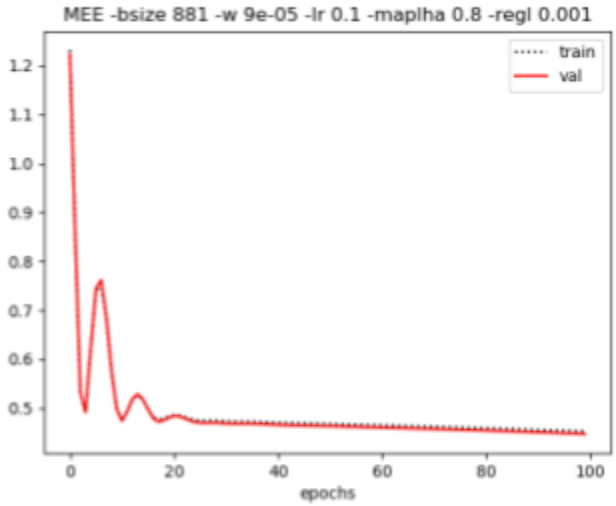

Figura A.8 MEE grid search fase 2

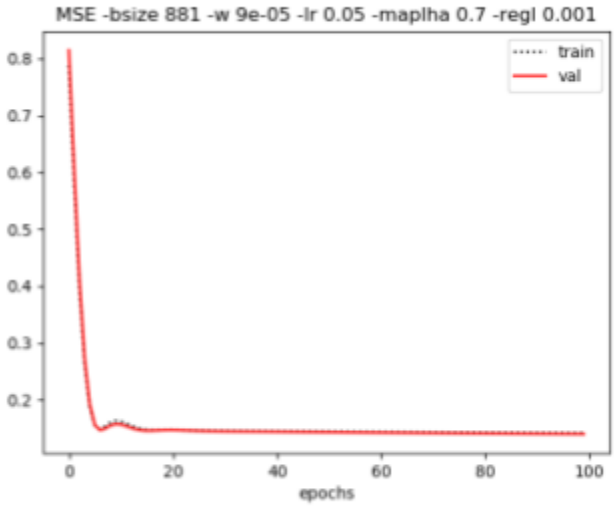

Figura A.9 MSE grid search fase 2

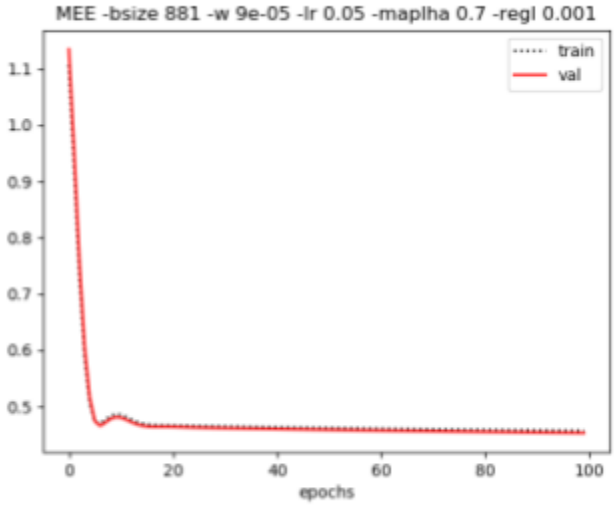

Figura A.10 MEE grid search fase 2

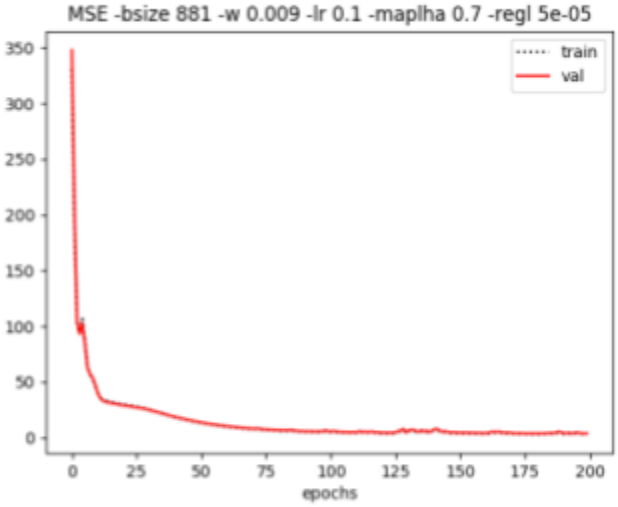

Figura A.11 MSE grid search fase 3

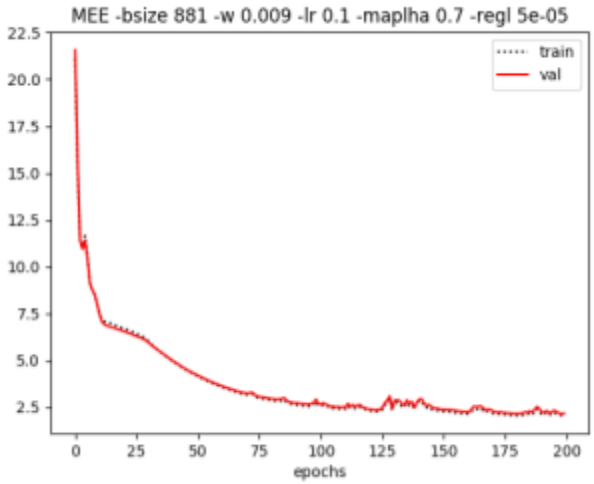

Figura A.12 MEE grid search fase 3