Rete Neurale

Alexandru Cretu (0324428)
Simone Nicosanti (0334319)
Gianmatteo Gabrielli (0333313)

2022/2023

Indice

1 Progetto

1.1 Contesto ed obiettivi

Il seguente progetto è stato realizzato con l'obiettivo di implementare una Rete Neurale per risolvere problemi di classificazione e regressione. (L. Bottou [2018], L. Grippo [2011]).
Il linguaggio scelto per tale implementazione è **Python**, poiché noto a tutti i membri del gruppo.

1.2 Requisiti per l'esecuzione

Per eseguire è necessario un ambiente **Python** da versione 3.9 in su, con installate le seguenti librerie:
- **pandas**;
- **numpy**;
- **matplotlib**;
- **scipy**;
- **sklearn**;

Per installarle, si può eseguire il seguente comando: *"pip3 install nomelibreria"*.

1.3 Data Set e Pre-elaborazione

I datasets utilizzati sono stati scaricati da **www.kaggle.com** ed opportunamente divisi in *Training Set*, *Validation Set* e *Test Set*.

La divisione è stata effettuata facendo lo shuffle del dataset, per poi suddividerlo nei diversi set elencati precedentemente, con le seguenti proporzioni di default:
- *Test Set* e *Validation Set* rappresentano ognuno il 15%;
- *Training Set* rappresenta il rimanente, ovvero il 70%;

Inoltre viene effettuato il drop delle variabili target prima dell'addestramento per poi utilizzarle in fase di valutazione dell'errore.
Per la gestione di feature categoriche, viene utilizzata una codifica di tipo *One-Hot*, con il supporto della libreria **sklearn**.
Tale operazione comporta un aumento del numero di feature del dataset, data la conversione da categoriche a numeriche.
Per l'addestramento, i dati del *Training Set* vengono standardizzati.
In fase di testing, il *Testing Set* viene standardizzato usando media e deviazione standard del *Training Set*.

1.4 Architettura della rete neurale

1.4.1 Numero di Layer e Numero di Neuroni

L'architettura di default della rete neurale consiste in 2 *layer*, ognuno dei quali contiene un numero di *neuroni* pari a:

$$2/3 \cdot featuresNumber + labelsNumber \tag{1}$$

un valore empirico che, come riportato in Krishnan [2021] si comporta bene in diverse situazioni.

1.4.2 Back Propagation

All'interno del file "NeuralNetwork.py" è stata implementata la *Back Propagation* cercando di ridurre quanto più possibile l'utilizzo di cicli for, sfruttando le operazioni matriciali offerte dalla libreria **numpy**, cercando di ottimizzare quanto più possibile i tempi di addestramento.

1.4.3 Regolarizzazione

Impostando il valore di λ_{L_1} si attiva la *L1-Regularization*.
Impostando il valore di λ_{L_2} si attiva la *Squared L2-Regularization*.
Bushaev [2018].

1.4.4 Cross Validation

Abilitando la *Cross Validation*, si può attivare la combinazione ottima di numero di *layer*, numero di *neuroni* e *parametri di regolarizzazione* per il problema scelto.
Essa può essere eseguita sia sequenzialmente, che utilizzando più thread.

1.4.5 Inizializzazione dei pesi

- L'inizializzazione dei pesi viene effettuata in modo diverso in base all'algoritmo di *learning rate* scelto:
  - *Uniform Initialization*: per **RMSProp**, **AdaGrad**;
  - *He Initialization*: per **Diminishing Stepsize**, **AdaDelta**, **ADAM**, **NADAM** (Yadav [2018]);
- Il *bias* viene inizializzato preliminarmente a zero in ogni livello.

1.4.6 Scelta del Learning Rate

L'algoritmo per la scelta del *learning rate* può essere scelto tra:
- **Diminishing Stepsize**;
- **AdaGrad**;
- **AdaDelta**;
- **RMSProp**;
- **ADAM**;
- **NADAM**;

1.4.7 Algoritmi

L'algoritmo utilizzato dalla Rete Neurale può essere scelto tra:
- **Stochastic Average Gradient Aggregation (SAGA)**;
- **Stochastic Gradient Descent (SGD)**;

Entrambi gli algoritmi utilizzano il *dynamic sampling* come metodo di *noise reduction*.
Inoltre, è stata implementata la possibilità di configurare la rete in modo da poter prendere più volte nella stessa epoca un punto già considerato (con conseguente cambio dei tempi di addestramento).

2 Risultati

L'accuratezza è calcolata come:
- *Media dei punti classificati correttamente sul totale*, per problemi di classificazione;
- *Errore quadratico medio*, per problemi di regressione;

2.1 Classificazione

2.1.1 Dataset: Cancer (2 classi)

Eseguendo la *Cross-Validation* su tutte le possibili combinazioni di:
- Numero di *Layer*: [2, 3];
- Numero di *Neuroni*: [22, 128, 256];
- *Parametro di regolarizzazione L1*: [0.001, 0.01, 0.1];
- *Parametro di regolarizzazione L2*: [0.001, 0.01, 0.1];

Si trova che la configurazione ottima per il problema è:

Layer	Neuroni	λ_{L_1}	λ_{L_2}
3	22, 128, 256	0.0	0.01

Algoritmo	Metodo	Epoche	Generalization Accuracy [%]	Tempo [min:s]
SGD	NADAM	10	0.973	0:01

2.1.2 Dataset: MNIST (10 classi)

Risultati ottenuti cambiando l'algoritmo per la scelta del *learning rate*:

Algoritmo	Metodo	Epoche	Generalization Accuracy [%]	Tempo [min:s]
SGD	DIMINISHING	5	0.850	1:07
SGD	ADAGRAD	5	0.894	1:07
SGD	RMSPROP	5	0.955	1:08
SGD	ADAM	5	0.948	1:09
SGD	NADAM	5	0.954	1:12

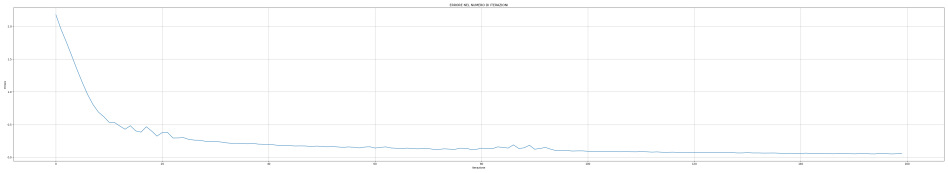

Figura 1: Grafico dell'errore relativo al MNIST.

2.1.3 Dataset: Chinese (15 classi)

Risultati ottenuti facendo variare il *parametro di regolarizzazione* λ_{L_2}:

Algoritmo	Metodo	λ_{L_2}	Epoche	Generalization Accuracy [%]	Tempo [min:s]
SGD	NADAM	0.001	25	0.796	8:38
SGD	NADAM	0.01	25	0.794	8:45
SGD	NADAM	0.1	25	0.792	8:28
SGD	NADAM	1.0	25	0.805	8:58
SGD	NADAM	10.0	25	0.681	9:06

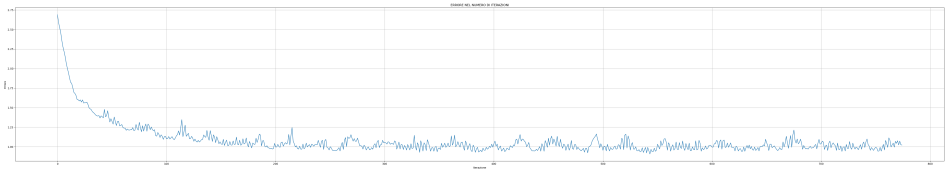

Figura 2: Grafico dell'errore relativo a Chinese.

2.2 Regressione

2.2.1 Dataset: Students

Risultati ottenuti facendo variare il numero di *epoche*:

Algoritmo	Metodo	Epoche	Generalization RMSE	Tempo [min:s]
SGD	NADAM	1	339.267	0:00.04
SGD	NADAM	5	55.904	0:00.20
SGD	NADAM	10	14.226	0:00.35
SGD	NADAM	50	4.09	0:01.78
SGD	NADAM	100	4.00	0:03.51

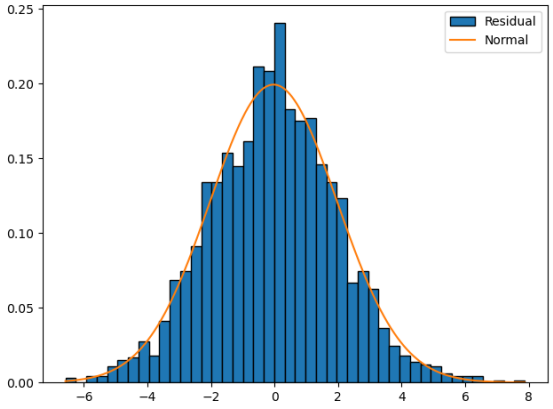

Figura 3: Istogramma dei Residui relativo a Students.

2.2.2 Dataset: Concrete

Risultati ottenuti cambiando l'algoritmo per la scelta del *learning rate* ed utilizzando il **SAGA** come algoritmo:

Algoritmo	Metodo	Epoche	Generalization RMSE	Tempo [min:s]
SAGA	DIMINISHING	1000	306.347	0:11:35
SAGA	ADAGRAD	1000	302.798	0:11.44
SAGA	ADADELTA	1000	102.301	0:45.79
SAGA	RMSPROP	1000	135.838	0:11.71
SAGA	ADAM	1000	165.665	0:12.10
SAGA	NADAM	1000	108.596	0:12.80

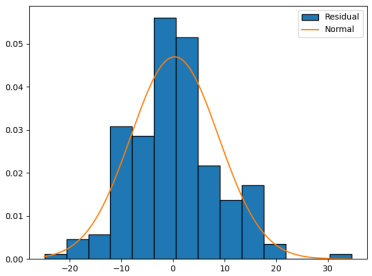

Figura 4: Istogramma dei Residui relativo a Concrete.

3 Bibliografia

Riferimenti bibliografici

J. Nocedal L. Bottou, F. Curtis. Optimization methods for large-scale machine learning. *Society for Industrial and Applied Mathematics*, 2018.

M. Sciandrone L. Grippo. *Metodi di Ottimizzazione Non Vincolata*. Springer, 2011.

S. Krishnan. How do determine the number of layers and neurons in the hidden layer. *Geek Culture*, 2021.

V. Bushaev. Adam — latest trends in deep learning optimization. *Towards Data Science*, 2018.

S. Yadav. Weight initialization techniques in neural networks. *Towards Data Science*, 2018.