Relazione Progetto Metodi di Ottimizzazione per Big Data

Matteo Scarcella

July 7, 2024

Indice

1 Introduzione

Il progetto ha avuto come obiettivo la creazione e l'addestramento di una rete neurale come modello di machine learning. Per testare la rete è stato scelto il dataset `mushroom.csv`, contenente informazioni sulla struttura fisica dei funghi e le rispettive etichette che indicano se il fungo è commestibile o velenoso. La rete neurale è stata addestrata per distinguere fra queste due classi, dunque il problema affrontato è di classificazione binaria. Nel seguito verranno illustrati i vari passaggi del progetto.

2 Preprocessing del Dataset

Il primo step è stato il trattamento e l'elaborazione dei dati.

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
0	1372	2	2	10	3.807467	1545	11	1.804273	1
1	1461	2	2	10	3.807467	1557	11	1.804273	1
2	1371	2	2	10	3.612496	1566	11	1.804273	1
3	1261	6	2	10	3.787572	1566	11	1.804273	1
4	1305	6	2	10	3.711971	1464	11	0.943195	1

Figure 1: Dataset originale

Dati i valori non omogenei, si è effettuata una normalizzazione per portare i valori delle features nell'intervallo $[0, 1]$, così da avere dati sulla stessa scala. Inoltre, la colonna delle etichette era originariamente di tipo binario, ed è stata eseguita su di essa la codifica one-hot encoding.

	cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class_0	class_1
0	0.725542	0.333333	0.333333	0.909091	0.992737	0.432894	0.916667	1.000000	0.0	1.0
1	0.772607	0.333333	0.333333	0.909091	0.992737	0.436257	0.916667	1.000000	0.0	1.0
2	0.725013	0.333333	0.333333	0.909091	0.941896	0.438778	0.916667	1.000000	0.0	1.0
3	0.666843	1.000000	0.333333	0.909091	0.987549	0.438778	0.916667	1.000000	0.0	1.0
4	0.690111	1.000000	0.333333	0.909091	0.967835	0.410199	0.916667	0.515404	0.0	1.0

Figure 2: Dataset pre-processato

Infine, si è suddiviso il dataset in due parti: un train set per l'addestramento e un test set per la valutazione.

3 Rete Neurale

3.1 Architettura

L'architettura della rete neurale progettata è composta da $L = 4$ strati:

- Input layer: 8 neuroni (corrispondenti alle 8 feature del dataset)
- Primo hidden layer: 100 neuroni
- Secondo hidden layer: 50 neuroni
- Output layer: 2 neuroni (corrispondenti alle due classi di output)

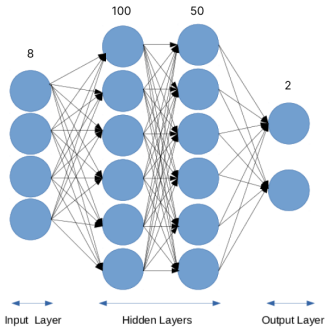

Figure 3: Architettura della rete neurale

Ogni strato nascosto utilizza la funzione di attivazione ReLU, mentre l'output layer utilizza la funzione di attivazione sigmoide. In accordo con l'utilizzo della funzione ReLU, per permettere alla rete di convergere più rapidamente è stata effettuata l'inizializzazione di He sui pesi, ovvero ogni peso è stato inizializzato a un valore casuale di una distribuzione normale con media nulla e deviazione standard pari a $\sqrt{\frac{2}{n^l}}$. Per valutare la performance della rete neurale, è stata scelta come funzione di loss la cross-entropy, ovvero:

$$-\sum_{k=1}^{p} y_k \log \varphi_k \tag{1}$$

dove y è il valore vero e φ è l'output stimato dalla rete.

3.2 Costruzione della Rete Neurale

Per costruire la rete neurale si è creata una classe `NeuralNetwork`. I metodi cardine della classe sono:

`forward_propagation(self,x)` che permette effettuare una predizione. Viene passato in ingresso un campione del dataset al fine di generare un output stimato tramite combinazioni lineari con le

matrici dei pesi. I risultati parziali vengono passati in ingresso alle funzioni di attivazione.

`back_propagation(self,y)` in cui si calcolano i gradienti dell'errore rispetto ai pesi.

4 Addestramento

Successivamente al passo di back-propagation, e quindi una volta calcolati tutti i gradienti, si procede all'aggiornamento dei pesi. Gli aggiornamenti possono essere effettuati tramite svariati metodi di ottimizzazione. In particolare, nel progetto ne sono stati implementati due: Stochastic Gradient Descent e Adam. Nel primo caso, le matrici dei pesi vengono aggiornate tramite la legge d'aggiornamento:

$$W_{k+1}^l = W_k^l - \alpha \frac{\partial E}{\partial W^l} \tag{2}$$

Per ogni strato l. Nel caso dell'algoritmo Adam invece le matrici dei pesi vengono aggiornate come segue:

$$W_{k+1}^l = W_k^l - \alpha \frac{\hat{m}_k}{\sqrt{\hat{v}_k} + \varepsilon} \tag{3}$$

Con:

$$\begin{aligned} \hat{m}_k^l &= \frac{m_k^l}{1-\beta_1^k} \\ \hat{v}_k^l &= \frac{v_k^l}{1-\beta_2^k} \\ m_{k+1}^l &= \beta_1 m_k^l + (1-\beta_1)\frac{\partial E}{\partial W^l} \\ v_{k+1}^l &= \beta_2 v_k^l + (1-\beta_2)\left(\frac{\partial E}{\partial W^l}\right)^2 \end{aligned} \tag{4}$$

Per ogni strato l.

L'addestramento della rete è stato eseguito sui dati preprocessati, utilizzando la funzione cross-entropy come metrica di errore. Il meccanismo utilizzato è stato il seguente:

```
Per ogni epoca:
    Per ogni campione:
        - Predizione dell'output
        - Calcolo dei gradienti
        - Aggiornamento dei pesi
```

Il criterio di arresto dell'addestramento è stato fissato a una loss inferiore a 0.1, unito ad un numero massimo di epoche fissato pari a 50, e inoltre è stata aggiunta la possibilità di arrestare prematuramente l'addestramento premendo la combinazione di tasti `CTRL+C`. In ognuno di questi casi, i pesi correnti della rete vengono salvati offline nei file `W.npy` e `b.npy`.

5 Valutazione delle Prestazioni

Utilizzando l'architettura delle rete suddetta, unita all'algoritmo SDG con un learning-rate $\alpha = 0.01$ la rete ha terminato l'addestramento dopo 14 epoche con una loss media pari a 0.0997. Per valutare

le prestazioni della rete, è stato creato un nuovo script che carica i pesi addestrati e predice gli output sul test set. I risultati ottenuti sono stati molto soddisfacenti, con una loss media di 0.128 (non molto distante dalla loss media sul set di addestramento) e un'accuracy score di 0.947. Usando invece l'algoritmo Adam, dopo alcuni tentativi sulla scelta dei parametri β_1 e β_2, l'algoritmo è giunto a convergenza in 7 epoche con $\beta_1 = 0.99$ e $\beta_2 = 0.999$, ovvero la metà rispetto a quelle impiegate da SDG. Testando i pesi ottenuti sui dati di test, i risultati sono ugualmente soddisfacenti, con una loss media di 0.118 e un'accuracy score di 0.956.