

Multilayered Perceptron - Classificazione Multiclasse

Lorenzo Baiardi

Giugno 27, 2022

1 Introduzione

In questa relazione verrà descritta l'architettura multilayered perceptron, una rete neurale che è in grado di apprendere dai dati forniti in input (features) fornendo un valore in uscita (prediction).

Il problema di classificazione (binaria) consiste ad esempio: data un'immagine di un animale, verificare che sia presente un cane o un gatto.

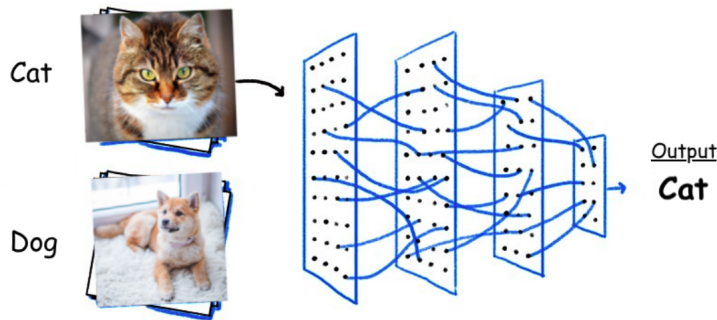


Figura 1: Esempio classificazione binaria

Nel caso di un problema multiclasse, il numero di oggetti da identificare è maggiore di due, di conseguenza la rete sarà in grado di classificare $k > 2$ classi.

2 MLP

Il Multilayer Perceptron è una struttura di rete basata sul perceptron, ma utilizzando più livelli.

Ogni livello è costituito da dei "neuroni" che hanno la funzione di Perceptron, cioè quello di fornire un valore in uscita dati dei valori di ingresso, il quale servirà per i successivi layer.

I layer possono essere distinti principalmente in tre categorie:

- Input Layer: composta dai nostri dati in ingresso. Il numero dei "neuroni" iniziali è dettato dal numero di features di cui abbiamo a disposizione.
- Hidden Layers: i layers "nascosti", dove effettivamente vengono processati i dati dai vari neuroni. Possono essere esserci più hidden layer in base al tipo di problema, e questi possono variare anche nel numero di neuroni.
- Output Layer: una volta che i dati vengono elaborati, la rete fornisce in uscita un predizione della categorizzazione.

Per i nostri esempi è sufficiente avere un unico hidden layer.

Esempio di Rete

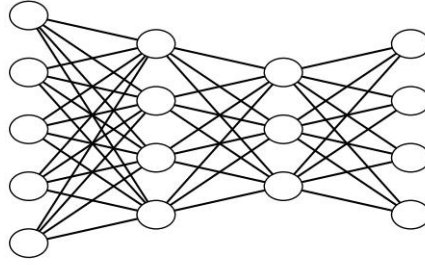


Figura 2: Esempio di Rete MLP Multiclasse

2.1 BackPropagation

Per l'apprendimento della rete utilizziamo l'algoritmo di backpropagation composto da due parti: una forward dove viene presentato un'osservazione alla rete e successivamente calcolato il valore di uscita con determinazione dell'errore, e una backward dove l'errore calcolato dalla forward viene propagata all'indietro verso tutti i layers. In quest'ultima fase i pesi di tutti i layer vengono aggiustati per ottimizzare la classificazione per le successive osservazioni.

Per l'ottimizzazione dei pesi: $w_{t+1} = w_t - \gamma * \nabla_w L^{(i)}$

Il parametro γ rappresenta il grado di apprendimento della rete (learning rate).

2.2 Inizializzazione dei Parametri

Nella fase d'inizializzazione di una rete è necessario inizializzare correttamente i parametri.

Non è possibile inizializzare i pesi della rete a zero, dato che in questo caso la rete si fermerebbe al primo calcolo del gradiente.

Esistono varie inizializzazioni, in questo elaborato ne vedremo in particolare due:

- Casuale: vengono randomizzati i valori iniziali
- Glorot: i valori saranno presi da distribuzione uniforme $U(-d, d)$ con $d = \sqrt{\frac{6}{\text{padri}_j + \text{figli}_j}}$

2.3 Funzioni di attivazione

- Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$
- Tanh: $\tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$
- Relu: $\text{Relu}(x) = \max(0, x)$

2.3.1 Softmax

Utilizziamo anche la funzione di attivazione Softmax che sarà necessaria per mappare il vettore delle varie predizioni in un vettore θ tale che: $\sum_j \theta_j = 1$

Softmax: $\theta_j = \frac{e^{v_j}}{\sum_r e^{v_r}}$ $r = 1, \dots, k$ con k = numero di classi

2.4 Multiclass Cross Entropy

Per il caso multiclasse utilizziamo la funzione di loss: $l(\theta, y) = -\frac{1}{n} \sum_{j=1}^k y_j \log \theta_j$ con n = osservazioni.

2.5 Metodo del gradiente stocastico

La parte critica dell'algoritmo backpropagation è il calcolo del gradiente, soprattutto quando il dataset è relativamente grande.

Per risolvere questo problema possiamo adottare diverse strategie: una di queste è l'SGD (Stochastic Gradient Descent) che è il metodo utilizzato all'interno dell'elaborato per il passaggio dei dati del dataset.

Preso un dataset, randomizziamo le varie osservazioni e da questo preleviamo un singolo indice i dal quale poi verrà calcolato la predizione e il suo relativo errore.

Successivamente, preleveremo un altro indice per rieseguire il calcolo, fino a che non completeremo il dataset. L'iterazione di tutto il dataset si chiama epoca.

Questo metodo permette di aumentare la velocità del calcolo del gradiente rispetto al Gradient Descent, cioè il passaggio dell'intero dataset.

Un ultimo metodo che è possibile utilizzare è il Minibatch che consiste nell'iterazione di un range di indici (range = batch-size) alla volta fino al completamento del dataset.

Se il range che utilizziamo per il minibatch è uguale al numero di tutto il dataset, otteniamo il Gradient Descent, se invece è uguale a uno riotteniamo il metodo del gradiente stocastico.

L'SDG è più efficiente del gradient descent perché ottimizza i tempi dovuti al calcolo del gradiente, ma con lo svantaggio di essere meno preciso nella valutazione dell'errore. Il minibatch è il metodo che riesce a coniugare una migliore precisione con una maggiore velocità (a discapito di una maggiore implementazione).

3 Dataset

I dataset utilizzati sono stati presi dal sito: <https://archive-beta.ics.uci.edu/>

Entrambi i dataset hanno un numero di classi maggiori di 3, e numero di osservazioni maggiori di 5000.

Il primo dataset utilizzato consiste nell'individuare, dati i features, se una macchina è soggetta a qualche tipo di malfunzionamento:

- numero di features = 5
- numero di classi = 4
- numero di osservazioni = 10000

Il secondo dataset è una classificazione di rane in base alle caratteristiche fornite in input:

- numero di features = 21
- numero di classi = 10
- numero di osservazioni = 7195

Di queste osservazioni ne vengano estrapolate circa mille per il validation set.

4 Valori Attesi

Per come è stata progettata la nostra architettura, ci aspetteremo che la rete abbia una accuratezza peggiore nelle prime iterazioni dato che necessita di informazioni per l'apprendimento. Nelle successive iterazioni, la rete sarà in grado di apprendere molto di più migliorando quindi la precisione nella classificazione. Più sono i dati e il numero di epoche, più la rete sarà accurata nella classificazione.

5 Risultati

- Dataset 1: Fallimento delle macchine.
Sono stati presi 9000 osservazioni per 50 epoche per l'apprendimento della rete, con un learning rate pari a 0.0001.
 - Apprendimento:

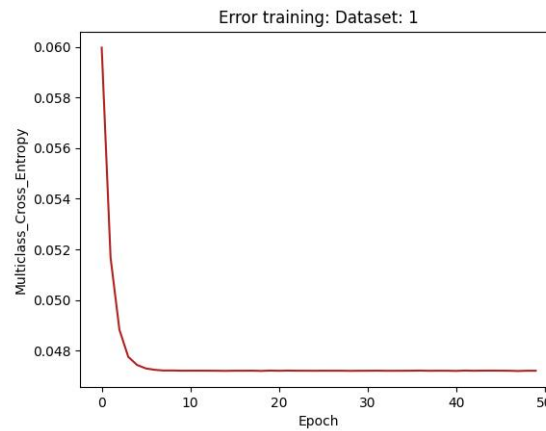


Figura 3: Multiclass Cross Entropy sull'errore di training

Il valore di predizione inizialmente si rivela già notevolmente basso già dalle prime iterazioni, fino ad a convergere.

- Predizioni:
Le restanti osservazioni, circa 1000, sono state utilizzate per il validation set.

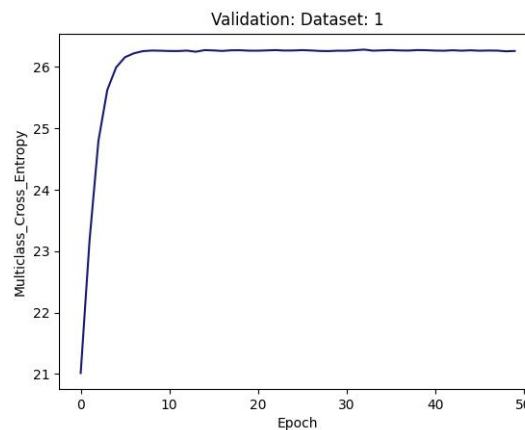


Figura 4: Validation set sulle osservazioni

- Dataset 2: Classificazione delle rane.
Sono stati presi 6000 osservazioni per 50 epoche per l'apprendimento della rete, con learning rate pari a 0.0001.
 - Apprendimento:

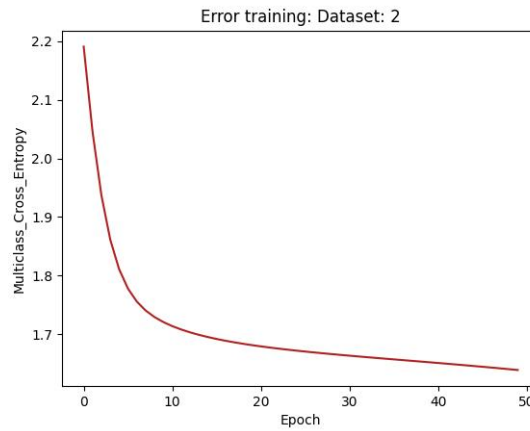


Figura 5: Multiclass Cross Entropy sull'errore di training

Come è possibile vedere, in questo caso, il valore iniziale di loss è molto alto fino a convergere ad un valore più basso.

- Predizione:
Le restanti osservazioni, circa 1000, sono state utilizzate per il validation set.

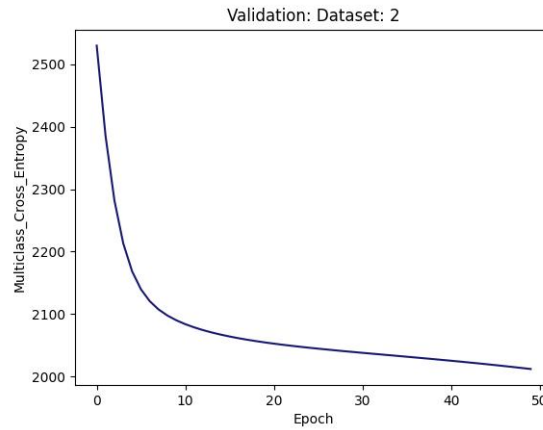


Figura 6: Validation set sulle osservazioni

6 Conclusioni

Come abbiamo potuto osservare dai grafici, la rete una volta che apprende dai dati riesce a sbagliare sempre meno fino ad arrivare al punto di convergenza. Nel dataset due l'errore inizialmente è più alto del dataset uno, ma con il passare delle iterazioni anche queste tende alla convergenza come è da previsione. Questo è probabilmente dovuto dal numero di epoche o dalla numero di osservazioni del secondo dataset.

7 Caratteristiche Terminale

Il terminale su cui sono state svolte le prove:

- Sistema Operativo: Windows 11 PRO
- Processore: Intel I5-8600K
- Scheda Video: Nvidia GEFORCE 1050-Ti
- Ram: 16GB DDR4 3600MHz

8 Riferimenti e Letture

- [backpropagation](#)
- [archivio datasets](#)
- [metodo per il plot della rete](#)