

3 Multi-Layer Perceptron e Approssimazione di Funzioni

Come discusso in precedenza, un **Multi-Layer Perceptron (MLP)** è una rete neurale in avanti con uno o più ingressi reali, degli strati nascosti, ognuno dei quali è caratterizzato da un numero specifico di neuroni, e una o più uscite reali. Per semplificare la notazione, ottenendo comunque risultati interessanti, conviene concentrarsi su MLP con un solo strato nascosto e una sola uscita. In questa ipotesi, si parla di $(n, m, 1)$ -MLP per indicare un MLP con $n \in \mathbb{N}_+$ ingressi, un solo strato nascosto caratterizzato da $m \in \mathbb{N}_+$ neuroni e un'uscita. Si noti che, in analogia a quanto fatto per i SLP, i valori di bias vengono gestiti mediante neuroni aggiuntivi nello strato di ingresso e nell'unico strato nascosto. In Figura 7 viene mostrato un $(3, 4, 1)$ -MLP.

Si consideri un $(n - 1, m - 1, 1)$ -MLP caratterizzato da una funzione di attivazione $g : \mathbb{R} \rightarrow \mathbb{R}$ utilizzata in tutte le unità di McCulloch-Pitts. L'obiettivo è usare il MLP come un approssimatore di una funzione $f : \mathbb{R}^n \rightarrow \mathbb{R}$ di cui si conoscano i valori per alcuni vettori. Come nel caso del SLP, alcuni dei vettori in cui la funzione è nota vengono raccolti in un training set, mentre i rimanenti vengono raccolti in un test set. Per semplificare la notazione, senza comunque perdere di generalità, si ipotizza che tutti i vettori $\mathbf{x} \in \mathbb{R}^n$ utilizzati come ingressi del MLP, compresi quelli raccolti nel training set e nel test set, siano del tipo

$$\mathbf{x} = (x_1, x_2, \dots, x_{n-1}, x_n = -1) \quad (24)$$

dimensione vettori di input

In più, si ipotizza che tutti i vettori $\mathbf{y} \in \mathbb{R}^m$ che vengono posti in ingresso al neurone di uscita siano del tipo

$$\mathbf{y} = (y_1, y_2, \dots, y_{m-1}, y_m = -1) \quad (25)$$

dimensione dei vettori dello strato intermedio

Naturalmente, se si vuole effettivamente disporre di n valori di ingresso per il MLP, sarà sufficiente considerare un MLP con $n + 1$ neuroni di ingresso. Allo stesso modo, se si vuole disporre effettivamente di m valori in ingresso al neurone di uscita, sarà sufficiente considerare uno strato nascosto composto da $m + 1$ neuroni.

Sotto queste ipotesi, se $\mathbf{w}_j \in \mathbb{R}^n$ è il vettore dei pesi associati agli archi che collegano lo strato di ingresso con il j -esimo neurone dello strato nascosto, con $1 \leq j \leq m - 1$, sarà

$$y_j = g(\mathbf{w}_j \cdot \mathbf{x}) \quad (26)$$

vetture dei pesi che collegano lo strato di ingresso a j -esimo neurone dello strato nascosto
f.ne attivazione
+ output tra vettore primo strato e strato nascosto

In più, se $\mathbf{v} \in \mathbb{R}^m$ è il vettore dei pesi associati agli archi che collegano lo strato nascosto con il neurone di uscita, l'uscita della rete sarà

$$o(\mathbf{w}, \mathbf{x}) = g(\mathbf{v} \cdot \mathbf{y}) \quad (27)$$

vetture dei pesi associati tra strato nascosto e l'uscita.
+ output di rete
vetture ottenute da tutti gli y_j di prima.

dove è stato introdotto il **vettore dei pesi del MLP** $\mathbf{w} \in \mathbb{R}^{m+n \cdot (m-1)}$. Questo vettore dei pesi viene ottenuto concatenando gli m pesi di \mathbf{v} con gli n pesi \mathbf{w}_1 , gli n pesi di \mathbf{w}_2 e procedendo così fino ad arrivare agli n pesi di \mathbf{w}_{m-1} . Questo vettore raccoglie tutti i pesi del MLP e viene usato unicamente per raggruppare in modo ordinato tutti i pesi del MLP.

Quindi, scrivendo per esteso i prodotti scalari,

$$o(\mathbf{w}, \mathbf{x}) = g \left(\sum_{j=1}^m v_j y_j \right) = g \left(\sum_{j=1}^{m-1} v_j g \left(\sum_{i=1}^{n-1} w_{j,i} x_i - w_{j,n} \right) - v_m \right) \quad (28)$$

dove $w_{j,i}$, con $1 \leq j \leq m - 1$ e $1 \leq i \leq n$, è la i -esima componente del vettore \mathbf{w}_j , che non è altro che il vettore che collega lo strato di ingresso con il j -esimo neurone dello strato nascosto. Utilizzando la nomenclatura dei tensori, è possibile dire che $W = (\mathbf{w})_{j=1}^{m-1}$ è il 2-tensore che caratterizza il calcolo svolto dallo strato nascosto del MLP.

voglio esplicitare da cosa dipende o:
da tutti gli x , da tutti i componenti del vettore v ,
da tutti i w_j e di conseguenza da tutti i componenti di
ognuno dei w_j .

matrice composta da
 $m-1$ vettori colonna.
Ogni vettore colonna rappresenta i pesi delle
connessioni tra un neurone dello strato precedente
e tutti i neuroni dello strato nascosto.

Stessa logica di utilizzo: vogliamo approssimare una f.n.e (in questo disegno da \mathbb{R}^3 a \mathbb{R}), per farlo avremo un training set con i punti e i valori della f.n.e, per verificare la nostra bravura avremo un test set.

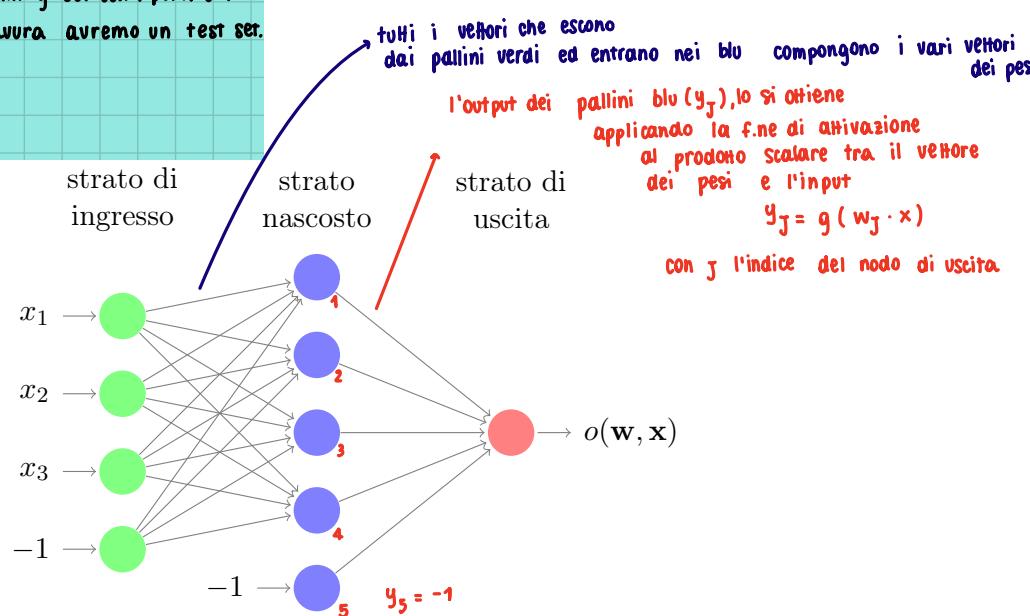


Figura 7: Esempio di un $(3, 4, 1)$ -MLP

Fissato un vettore di ingresso $\mathbf{x} \in \mathbb{R}^n$ del training set, l'errore commesso dal MLP nell'approssimare la funzione f per lo specifico \mathbf{x} vale

$$\epsilon(\mathbf{w}, \mathbf{x}) = f(\mathbf{x}) - o(\mathbf{w}, \mathbf{x}) \quad (29)$$

differenza tra
valore della f.n.e in \mathbf{x}
e l'output di rete

Introducendo l'errore quadratico e ipotizzando che $g \in C^\infty(\mathbb{R})$, esattamente com'è stato fatto per il SLP,

$$e(\mathbf{w}, \mathbf{x}) = \frac{1}{2} \epsilon^2(\mathbf{w}, \mathbf{x}) \quad (30)$$

errore quadratico errore commesso espresso ϵ , vogliamo cercare il gradiente di ϵ al variare dei w .
Fissato x , avremo $f(x)$ dal training set e varia unicamente al variare di w . Le componenti di w sono tutti i pesi che andiamo a mettere nel MLP.

si può cercare un vettore dei pesi del MLP utilizzando l'algoritmo di discesa del gradiente nel tentativo di minimizzare l'errore quadratico.

Nell'ipotesi che sia stato fissato un vettore di ingresso $\mathbf{x} \in \mathbb{R}^n$ del training set, allora l'uscita del MLP, l'errore e l'errore quadratico dipendono solo dal vettore dei pesi del MLP.

Si consideri il peso v_j , con $1 \leq j \leq m$, associato all'arco che collega il j -esimo neurone dello strato nascosto con il neurone di uscita

andiamo a calcolare la derivata dell'errore rispetto a v_j

$$\frac{\partial e}{\partial v_j}(\mathbf{w}) = \frac{1}{2} \cdot 2 \cdot \epsilon(\mathbf{w}) \frac{\partial \epsilon}{\partial v_j}(\mathbf{w}) \quad (31)$$

ma

$$\frac{\partial \epsilon}{\partial v_j}(\mathbf{w}) = -\frac{\partial o}{\partial v_j}(\mathbf{w}) \quad (32)$$

derivata dell'errore commesso ($\rightarrow \epsilon = f(\mathbf{x}) - o(\mathbf{w}, \mathbf{x})$)
costante

perché $f(\mathbf{x})$ non dipende dal vettore dei pesi del MLP. Ricordando ora la definizione di $o(\mathbf{w})$, si ottiene

$$\frac{\partial o}{\partial v_j}(\mathbf{w}) = g'(\mathbf{v} \cdot \mathbf{y}) \frac{\partial (\mathbf{v} \cdot \mathbf{y})}{\partial v_j} = g'(\mathbf{v} \cdot \mathbf{y}) y_j \quad (33)$$

e quindi, coerentemente con il fatto che, rispetto allo strato nascosto, l'uscita del MLP si comporta come un SLP, è possibile esprimere la seguente regola per l'aggiornamento dei pesi che collegano lo strato nascosto con l'uscita

Prima formula di aggiornamento

$$v_j \leftarrow v_j + \alpha \frac{(f(\mathbf{x}) - g(\mathbf{v} \cdot \mathbf{y}))}{\text{Coef. di apprendimento}} g'(\mathbf{v} \cdot \mathbf{y}) y_j \quad (34)$$

output di rete derivata dell'output di rete

per $1 \leq j \leq m$ e assumendo che l'ultima componente di \mathbf{y} valga -1 . Si noti che questa regola di aggiornamento dei pesi è identica a quella trovata per i SLP.

Ma cosa succede se andiamo a considerare non la variabile di j , ma una delle variabili raggruppate in uno dei vettori dei pesi che abbiamo per collegare l'input con uno dei nodi dello strato nascosto?

Questa regola consente di modificare i pesi che collegano lo strato nascosto con il neurone di uscita in modo da cercare un minimo locale dell'errore commesso dal MLP quando riceve in ingresso il vettore \mathbf{x} . Si noti che prima viene applicato il MLP all'ingresso \mathbf{x} per calcolare il vettore \mathbf{y} e viene applicata in direzione opposta la regola di aggiornamento dei pesi. Questo andamento dell'elaborazione, prima in avanti attraversando il MLP e poi all'indietro, sempre attraversando lo stesso MLP, è caratteristico dell'addestramento supervisionato applicato agli MLP.

Sempre nell'ipotesi che sia stato fissato un vettore di ingresso $\mathbf{x} \in \mathbb{R}^n$, si consideri ora $\mathbf{w}_j \in \mathbb{R}^n$, che è il vettore dei pesi associati agli archi che collegano lo strato di ingresso con il j -esimo neurone dello strato nascosto, e quindi $1 \leq j \leq m - 1$. La derivata dell'errore quadratico rispetto alla i -esima componente di \mathbf{w}_j vale

$$\frac{\partial e}{\partial w_{j,i}}(\mathbf{w}) = \frac{1}{2} \cdot 2 \cdot \epsilon(\mathbf{w}) \frac{\partial \epsilon}{\partial w_{j,i}}(\mathbf{w}) \quad (35)$$

w_j è il vettore dei pesi che collega l'input con quel nodo i . Fissato j abbiamo un vettore, fissato i , al suo interno abbiamo la variabile

Le v sono state messe a posto prima, qua si ottiene una formula di aggiornamento

ma, come già discusso, nascondendo che consideriamo nodo dello strato

$$\frac{\partial \epsilon}{\partial w_{j,i}}(\mathbf{w}) = -\frac{\partial o}{\partial w_{j,i}}(\mathbf{w}) \quad (36)$$

Le formule di aggiornamento non sono una sola ma sono tante quante gli strati di cui si calcola l'uscita. perché $f(\mathbf{x})$ non dipende dal vettore dei pesi del MLP. Ora, ricordando la definizione di $o(\mathbf{w})$, si ottiene

$$\frac{\partial o}{\partial w_{j,i}}(\mathbf{w}) = g'(\mathbf{v} \cdot \mathbf{y}) \frac{\partial (\mathbf{v} \cdot \mathbf{y})}{\partial w_{j,i}} \rightarrow \begin{array}{l} \text{da risolvere perché } \mathbf{v} \text{ non dipende} \\ \text{da } \mathbf{w}, \text{ ma } \mathbf{y} \text{ sì} \end{array} \quad (37)$$

y è il valore che si ottiene applicando la regola di McCulloch Pitts per il vettore dei pesi giusto

ma ora, contrariamente a quanto accaduto precedentemente, è necessario considerare la dipendenza del vettore \mathbf{y} dal vettore dei pesi del MLP

$$\frac{\partial (\mathbf{v} \cdot \mathbf{y})}{\partial w_{j,i}} = \mathbf{v} \cdot \frac{\partial \mathbf{y}}{\partial w_{j,i}} = \sum_{k=1}^m v_k \frac{\partial y_k}{\partial w_{j,i}} = \sum_{k=1}^{m-1} v_k \frac{\partial g(\mathbf{w}_k \cdot \mathbf{x})}{\partial w_{j,i}} \quad (38)$$

da cui

$$\frac{\partial (\mathbf{v} \cdot \mathbf{y})}{\partial w_{j,i}} = \sum_{k=1}^{m-1} v_k g'(\mathbf{w}_k \cdot \mathbf{x}) \frac{\partial (\mathbf{w}_k \cdot \mathbf{x})}{\partial w_{j,i}} \quad (39)$$

ma

$$\frac{\partial (\mathbf{w}_k \cdot \mathbf{x})}{\partial w_{j,i}} = \sum_{r=1}^n x_r \frac{\partial w_{k,r}}{\partial w_{j,i}} = \begin{cases} x_i & \text{se } r = i, k = j \\ 0 & \text{altrimenti} \end{cases} \quad (40)$$

e quindi

$$\frac{\partial (\mathbf{v} \cdot \mathbf{y})}{\partial w_{j,i}} = v_j g'(\mathbf{w}_j \cdot \mathbf{x}) x_i \quad (41)$$

Quindi, in sintesi, la regola di aggiornamento dei pesi $w_{j,i}$ associati agli archi che collegano i neuroni di ingresso con il j -esimo neurone dello strato nascosto è

$$\text{Seconda formula di aggiornamento} \quad w_{j,i} \leftarrow w_{j,i} + \alpha (f(\mathbf{x}) - g(\mathbf{v} \cdot \mathbf{y})) g'(\mathbf{v} \cdot \mathbf{y}) v_j g'(\mathbf{w}_j \cdot \mathbf{x}) x_i \quad (42)$$

con $1 \leq j \leq m - 1$ e $1 \leq i \leq n$.

$$\downarrow \frac{\partial o}{\partial w_{j,i}}(\mathbf{w}) \quad \downarrow \frac{\partial (\mathbf{v} \cdot \mathbf{y})}{\partial w_{j,i}}$$

Come si evince dalla regola di aggiornamento dei pesi tra l'ingresso e lo strato nascosto, l'errore compiuto dal MLP viene distribuito tra i pesi in modo simile a come viene distribuito nei pesi associati agli archi che collegano lo strato nascosto con l'uscita del MLP. Infatti, riassumendo quanto ottenuto, le regole di aggiornamento dei pesi sono

$$\mathbf{v} \leftarrow \mathbf{v} + \alpha \epsilon(\mathbf{w}) g'(\mathbf{v} \cdot \mathbf{y}) \mathbf{y} \quad (43)$$

$$\mathbf{w}_j \leftarrow \mathbf{w}_j + \alpha \epsilon(\mathbf{w}) g'(\mathbf{v} \cdot \mathbf{y}) v_j g'(\mathbf{w}_j \cdot \mathbf{x}) \mathbf{x} \quad (44)$$

Le due regole di aggiornamento dei pesi si esprimono così, ma non vanno calcolate in questo modo, perché la seconda formula dipende da v_j e sono quelli che abbiamo usato prima di applicare l'aggiornamento della regola di v .

In realtà quello che c'è da fare è 1) far passare l'input e produrre l'output (calcolo l'ingresso, calcolo del valore sui nodi dello strato intermedio e quindi calcolo gli y , una volta ottenuti gli y calcolo l'uscita [mi serve per calcolare l'errore]) 2) a questo punto, siccome stiamo ragionando per un x nel training set, sappiamo calcolare ϵ ; a questo punto x, y, ϵ li abbiamo, quindi la seconda formula dipende unicamente dal nodo j : cominciamo e facciamo variare i nodi dello strato intermedio uno dopo l'altro con j , da 1 a $m-1$; fissiamo $j=1$ quello che calcoliamo è un aggiornamento di w che dipenderà da x, w attuali, v attuali; il resto lo abbiamo. Facciamo questa operazione e aggiorniamo tutti i w .

Una volta che abbiamo aggiornato i w , possiamo andare ad aggiornare v , perché a questo punto v non dipende più dai w .

e quindi l'errore viene ripartito con coefficiente pari a 1 per l'aggiornamento del vettore dei pesi v e con coefficiente pari a $g'(v \cdot y)v_j$ per l'aggiornamento del vettore dei pesi w_j , con $1 \leq j \leq m-1$. Si noti che le formule di aggiornamento dei pesi di un MLP vengono spesso espresse nell'ipotesi che la funzione di attivazione g sia una funzione logistica in modo da potere sfruttare la proprietà della derivata di questa funzione già discussa nel caso del SLP. Sotto questa ipotesi, valgono le seguenti regole di aggiornamento dei pesi

$$\text{Se la f.n. di attivazione è } S(x) = \frac{1}{1 + e^{-x}} \quad \left\{ \begin{array}{l} v \leftarrow v + \alpha (f(x) - o(w, x)) o(w, x) (1 - o(w, x)) y \xrightarrow{\text{dallo strato nascosto all'uscita}} \\ w_j \leftarrow w_j + \alpha (f(x) - o(w, x)) o(w, x) (1 - o(w, x)) v_j y_j (1 - y_j) x \end{array} \right. \quad (45)$$

e la sua derivata:

$$S'(x) = S(x)(1 - S(x)) \quad \text{Infine, si noti come le regole di addestramento dei pesi ottenute per un MLP con uno strato}$$

i pesi w_j che sono l'ingresso al j -esimo nodo dello strato nascosto

nascosto e una sola uscita abbiano una struttura caratteristica che è comune a tutti gli MLP, qualsiasi sia il numero di strati nascosti e di uscite. Per prima cosa è necessario calcolare i valori delle uscite dei neuroni nei vari strati prima di procedere ad aggiornare i pesi. Infatti, prima si applica il MLP al vettore di ingresso per ottenere l'uscita mediante un processo detto **forward propagation**. Quindi, si valuta l'errore e si ripartisce sui vari strati l'errore commesso, uno strato alla volta, mediante un processo detto **backward propagation**. Per questa sua caratteristica, l'algoritmo di addestramento che prevede di minimizzare l'errore quadratico compiuto dal MLP sui vettori del training set prende il nome di **algoritmo di backpropagation**. L'algoritmo di backpropagation per un $(n-1, m-1, 1)$ -MLP è mostrato nell'Algoritmo 3.

Algoritmo 3 Algoritmo di addestramento supervisionato per un $(n-1, m-1, 1)$ -MLP

```

function MLP _ LEARN( $T, \alpha, \delta, \rho$ )*
    randomize  $v \in \mathbb{R}^m$                                  $\triangleright$  inizializza  $v$ 
    for  $1 \leq j \leq m-1$  do
        randomize  $w_j \in \mathbb{R}^n$                        $\triangleright$  inizializza  $w_j$ 
    end for
    for  $1 \leq r \leq \rho$  do                                 $\triangleright$  ripeti per non più di  $\rho$  epochhe
         $\bar{\epsilon} \leftarrow 0$                                  $\triangleright$  inizializza l'errore medio sul training set
        for  $(x, f(x)) \in T$  do                           $\triangleright$  per ogni vettore del training set
             $a \leftarrow v \cdot y$                              $\triangleright$  inizializza  $a \in \mathbb{R}$ 
            for  $1 \leq j \leq m-1$  do  $\triangleright$  per i nodi dello strato nascosto tranne quello di bias
                 $b \leftarrow w_j \cdot x$                          $\triangleright$  inizializza  $b \in \mathbb{R}$ 
                for  $1 \leq i \leq n$  do                       $\triangleright$  aggiorna i pesi usati nello strato nascosto
                     $w_{j,i} \leftarrow w_{j,i} + \alpha (f(x) - g(a)) g'(a) v_j g'(b) x_i$ 
                end for
            end for
            for  $1 \leq i \leq m$  do                       $\triangleright$  aggiorna i pesi usati nello strato di uscita
                 $v_i \leftarrow v_i + \alpha (f(x) - g(a)) g'(a) y_i$ 
            end for
             $\bar{\epsilon} \leftarrow \bar{\epsilon} + \frac{|f(x) - g(v \cdot y)|}{|T|}$        $\triangleright$  aggiorna l'errore medio sul training set
        end for
        if  $\bar{\epsilon} < \delta$  then                                 $\triangleright$  se l'errore medio è sufficientemente piccolo
            return  $w$                                  $\triangleright$   $w$  è il vettore dei pesi dell'MLP cercato
        end if
    end for
    return  $w$                                  $\triangleright$   $w$  è il vettore dei pesi dell'MLP cercato dopo  $\rho$  epochhe
end function
```

*

- T : il training set formato da coppie $(x, f(x))$;
- α : il coefficiente di apprendimento;
- δ : la tolleranza accettata sull'errore medio; e
- ρ : il numero massimo di epochhe.

Si noti che questo algoritmo ha gli stessi parametri dell'algoritmo di addestramento supervisionato trovato per i SLP. Infatti, l'algoritmo per i SLP non è altro che un caso particolare dell'algoritmo di backpropagation.

- ✓ **Esempio 2.** Un uso tradizionale degli MLP e dell'algoritmo di backpropagation è nel riconoscimento del testo manoscritto. Un esempio ormai ritenuto classico è l'utilizzo di un $(784, 800, 10)$ -MLP per riconoscere le cifre manoscritte opportunamente isolate in piccole immagini formate da 28×28 pixel. Queste immagini usano una scala di 256 grigi, come mostrato in Figura 2, e quindi possono essere direttamente elaborate dal MLP senza necessità di pre-elaborazione.



Figura 8: Esempi di cifre manoscritte riconosciute da un $(784, 800, 100)$ -MLP

Il risultato dell'elaborazione del MLP è un vettore di 10 numeri reali in cui la posizione di una delle componenti massime rappresenta la cifra riconosciuta. Quindi, ad esempio, se il risultato del MLP è un vettore in cui la componente massima è in posizione 3, allora è possibile dire che il MLP ha classificato l'immagine in ingresso come una cifra 3. Si noti che il risultato della classificazione del MLP non è univocamente definito se il vettore risultato ha più componenti massime.

L'addestramento del MLP utilizzato per questo compito di riconoscimento di cifre manoscritte avviene usando l'algoritmo di backpropagation con un training set formato da 60 000 immagini pre-classificate. La misura delle prestazioni del MLP ottenuto mediante l'algoritmo di backpropagation avviene con un test set formato da 10 000 immagini pre-classificate. Il risultato documentato in letteratura parla di un MLP con un errore di classificazione sul test set inferiore al 2%, coerente con il grafico mostrato in Figura 9.

Il seguente **teorema di approssimazione universale** consente di usare gli MLP come strumenti generali per l'approssimazione di funzioni continue. Si noti comunque che esiste una variante del teorema di approssimazione universale che esplicita come sia possibile utilizzare un MLP per approssimare con tolleranza piccola a piacere anche funzioni con un numero finito di discontinuità nel dominio considerato. Questo teorema non viene però discusso perché richiede una notazione più complessa di quella utilizzata in questo testo.

In sintesi, il seguente teorema di approssimazione universale può essere letto come segue. Fissata una certa tolleranza $\epsilon \in \mathbb{R}_+$ nell'approssimazione di una funzione f continua su un compatto, è possibile trovare un MLP che possa essere utilizzato come approssimatore della

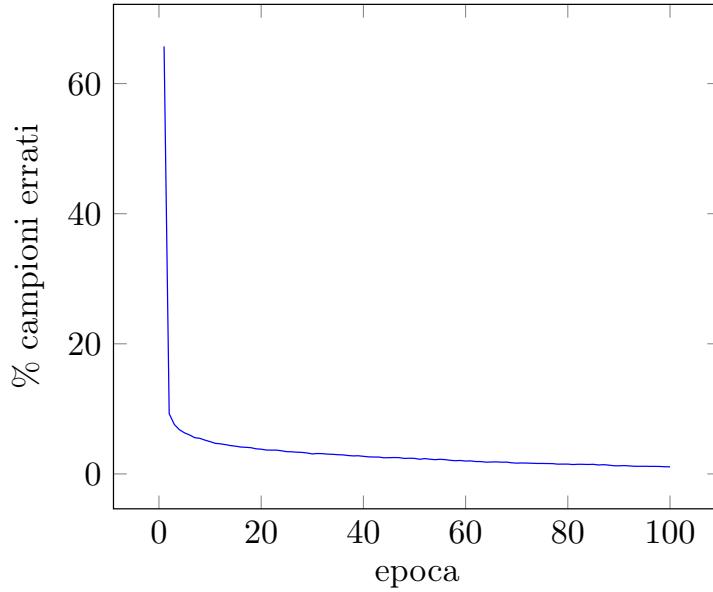


Figura 9: Andamento della percentuale di campioni errati nel training set in 100 epoche usando un $(784, 800, 100)$ -MLP

funzione f con la tolleranza richiesta. La struttura del MLP che è possibile utilizzare per l'approssimazione viene fissata dal teorema anche se, comunque, il teorema lascia notevoli gradi di libertà nella scelta dello specifico MLP.

Teorema 1 (Di Approssimazione Universale). Dato $n \in \mathbb{N}_+$, sia $D \subset \mathbb{R}^n$ compatto e sia $f : S \rightarrow \mathbb{R}$ una funzione reale di più variabili reali continua nell'aperto $S \subseteq \mathbb{R}^n$ con $D \subset S$. Sia $g : \mathbb{R} \rightarrow \mathbb{R}$ una funzione continua, monotona strettamente crescente e limitata. Per ogni $\epsilon \in \mathbb{R}_+$ esistono

- $m \in \mathbb{N}_+$
- v_j, θ_j , con $1 \leq j \leq m$
- $w_{j,i} \in \mathbb{R}$, con $1 \leq i \leq n$ e $1 \leq j \leq m$

tali che la funzione

$$h(\mathbf{x}) = \sum_{j=1}^m v_j g \left(\sum_{i=1}^n w_{j,i} x_i - \theta_j \right)$$

↓
approssimazione

J neuroni di McCulloch Pitts

bias

$$\max_{\mathbf{x} \in D} |f(\mathbf{x}) - h(\mathbf{x})| < \epsilon$$

soddisfi

In modo meno formale: se noi consideriamo una funzione continua in un compatto chiuso e limitato e consideriamo un MLP con 1 strato nascosto, i pesi del MLP li troviamo una volta che abbiamo fissato una tolleranza che ci interessa.
Fissiamo una tolleranza ϵ , questo teorema ci garantisce che esistono i pesi di un MLP che ci permette di approssimare, con una tolleranza ϵ , la funzione f su tutto il compatto D che abbiamo considerato.
Quindi MLP non è qualiasi: n ingressi, uno strato nascosto di cui non conosciamo il n. di neuroni, i neuroni dello strato nascosto usano una funzione g abbastanza generica, l'unico strato di uscita ha un unico neurone che ha una funzione di attivazione identità

(48)

Si noti che non è fissata la funzione di attivazione utilizzata per i neuroni dello strato nascosto. Viceversa, si assume che il MLP utilizzi una funzione di attivazione identità per il calcolo dell'uscita. Quest'ultima circostanza è del tutto ragionevole perché la scelta di una funzione di attivazione limitata avrebbe anche limitato a priori l'uscita del MLP e quindi avrebbe reso il MLP meno efficace nell'approssimare f , visto che non è stata posta una limitazione a priori ai valori di f .