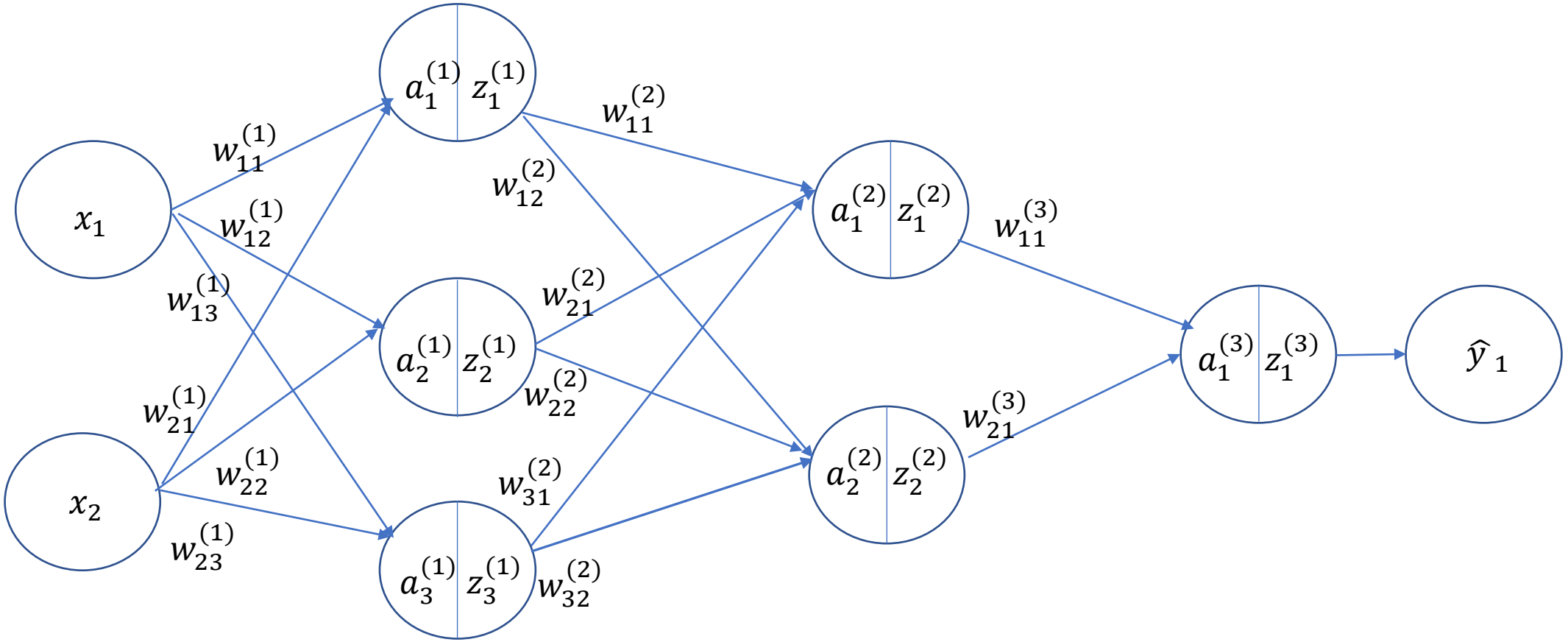
Backpropagation

MLP: Multi Layer Perceptron – Rete Feed Forward Fully connected



$d = 2, \quad L = 2 \qquad s = 1$

$z_i^{(0)} = x_i, \quad i = 1, \dots, d \qquad z_i^{(L+1)} = \hat{y}_i, \quad i = 1, \dots, s$

$$a_i^{(\ell)} = \sum_{j=0}^{N^{(\ell)}} w_{ji}^{(\ell)} z_j^{(\ell-1)} \qquad z_i^{(\ell)} = f(a_i^{(\ell)})$$

f: activation function: funzione **derivabile** non lineare.

Sia $\hat{y} = [\hat{y}_1, .., \hat{y}_s]$ l'output prodotto dalla rete in corrispondenza all'esempio di training $x = [x_1, x_2, .. x_d]$, e sia $y = [y_1, y_2, .., y_s]$ il vettore delle etichette corrispondente dell'esempio di training.
Scegliamo come loss function la somma dei quadrati degli errori tra l' output della rete e la corrispondente etichetta:

$$L(y, \hat{y}) = \frac{1}{2} \sum_{i=1}^{s} (y_i - \hat{y}_i)^2 = \frac{1}{2} \|y - \hat{y}\|_2^2$$

La dipendenza dai pesi w è implicita in $\hat{y}$.

L'errore $C(w)$ sull'intero training set è la media di $L(\mathbf{w},\mathbf{x})$ su tutti gli esempi appartenenti n_T al training set.

$$C(w) = \frac{1}{n_T} \sum_{j=1}^{n_T} L\left(y^{(j)}, \hat{y}^{(j)}\right)$$

dove $\hat{y}^{(j)} = \left[\hat{y}_1^{(j)}, .., \hat{y}_s^{(j)}\right]$ l'output prodotto dalla rete in corrispondenza al *j*-esimo esempio di training $x^{(s)} = [x_1^{(s)}, x_2^{(s)}, .. x_d^{(s)}]$, e sia $y = \left[y_1^{(j)}, y_2^{(j)}, .. y_s^{(j)}\right]$ il vettore delle etichette corrispondente dell'esempio di training.

Il problema dell'addestramento

La costruzione di una rete multistrato con d ingressi e s uscite consiste nello **scegliere l'architettura** (numero di strati e numero di neuroni di ogni strato), e nell'addestrare la rete, ossia nel determinare il vettore $w \in \mathbb{R}^m$ le cui componenti corrispondono ai parametri incogniti (pesi e soglie dei neuroni nei vari strati).

La scelta dei parametri, in addestramento di tipo **supervisionato**, per una architettura fissata, viene in genere effettuata definendo un opportuno sottoinsieme dei dati disponibili etichettati

$$T = \{(x^{(j)}, y^{(j)}), x^{(j)} \in \mathbb{R}^d, y^{(j)} \in \mathbb{R}^s, j = 1, \dots, n_T\},$$

che costituisce il **training set** e risolvendo successivamente un problema di ottimizzazione del tipo:

$$\arg\min_{w} C(w) = \frac{1}{n_T} \sum_{j=1}^{n_T} L\left(y^{(j)}, \hat{y}^{(j)}\right)$$

in cui $L\left(y^{(j)}, \hat{y}^{(j)}\right)$ è il termine di errore relativo al j–mo campione di training e misura la distanza tra l'uscita desiderata $y^{(j)}$ e l'uscita $\hat{y}^{(j)}$ fornita dalla rete. La misura più usata è l'errore quadratico ma è possibile considerare anche funzioni di errore di struttura diversa. Nel seguito ci limiteremo a supporre che E sia una funzione continuamente differenziabile.

Scopo dell'addestramento non è quello di interpolare i dati di training, quanto piuttosto quello di modellare il processo che ha generato i dati.

Ciò implica che:

- la scelta dell'architettura,
- **la selezione dei dati da includere in T,**
- la definizione di L
- e la strategia di addestramento

devono tener conto dell'esigenza di assicurare buone capacità di generalizzazione.

Dal punto di vista teorico, uno dei problemi più importanti è quello di definire opportunamente la complessità del modello, e quindi il numero di parametri liberi, in relazione ai dati disponibili. Per le reti multistrato, a partire dai risultati sulla teoria statistica dell'apprendimento sono state stabilite delle stime del numero minimo dei campioni di training occorrenti per addestrare una rete in modo tale che si abbia una "buona" capacità di generalizzazione.

In pratica, tuttavia le stime teoriche possono essere inadeguate ed **occorre basarsi su opportune euristiche per la scelta della struttura e la definizione del training set**. In linea di massima, nel caso delle reti multistrato, vengono seguite due strategie fondamentali.

La **prima**, chiamata ***stabilizzazione strutturale*** consiste nello scegliere il numero di unità , attraverso **l'addestramento di una sequenza di reti in cui viene fatto crescere (o diminuire) il numero di neuroni**.

- Per ciascuna di tali reti i parametri vengono determinati minimizzando l'errore sul training set e le prestazioni delle varie reti sono confrontate attraverso una tecnica di *cross-validation*, valutando l'errore che ogni rete commette su un altro insieme di dati (*validation set*) non inclusi in *T*. **La rete selezionata è quella che fornisce l'errore minimo sul validation set.**

Le prestazioni di una rete addestrata vengono valutate utilizzando un terzo insieme di dati denominato *test set*, che non deve essere stato utilizzato né per la scelta del l'architettura, né per la determinazione dei parametri.

La **seconda strategia** si basa su una tecnica di regolarizzazione e consiste nell'aggiungere alla funzione di errore un termine di penalità sulla norma di w che ha l'effetto di restringere l'insieme entro cui vengono scelti i parametri. Ciò equivale, essenzialmente, ad imporre condizioni di regolarità sulla classe di funzioni realizzata dalla rete. L' addestramento della rete viene effettuato definendo la nuova funzione obiettivo

$$\arg\min_{w} C(w) = \frac{1}{n_T}\sum_{j=1}^{n_T} L\left(y^{(j)}, \hat{y}^{(j)}\right) + \gamma\|w\|_2^2$$

con $\gamma > 0$.

Il valore "ottimale" di γ può essere determinato utilizzando, anche in questo caso, una tecnica di cross-validation. In particolare, in corrispondenza a differenti valori di γ, si addestrano varie reti minimizzando la funzione d'errore C e viene successivamente prescelto il valore di a cui corrisponde il minimo errore sul validation set.

In alternativa alla tecnica di regolarizzazione, una strategia euristica talvolta utilizzata è quella **di interrompere prematuramente la minimizzazione (early stopping) della funzione d'errore**.

Questa tecnica si basa sull'idea di valutare periodicamente, durante il processo di minimizzazione, l'errore che la rete commette su un validation set ausiliario.

In generale, nelle prime iterazioni l'errore sul validation set diminuisce con la funzione obiettivo, mentre può aumentare se l'errore sul training set diviene "sufficientemente piccolo".

Il processo di addestramento termina quindi quando l'errore sul validation set inizia ad aumentare, perchè ciò potrebbe evidenziare l'inizio della fase di overfitting della rete, cioè della fase in cui la rete tende a interpolare i dati di training a scapito della capacità di generalizzazione.

Quale che sia la strategia di addestramento seguita, pur non essendo possibile ridurre banalmente i problemi di addestramento alla soluzione di un problema di ottimizzazione, è necessario, in pratica, **ripetere la minimizzazione in corrispondenza a varie architetture o a varie funzioni di errore**. La disponibilità di **algoritmi efficienti di ottimizzazione costituisce, quindi, uno strumento essenziale per la costruzione delle reti neurali**. La minimizzazione dell'errore di training C è , in generale, un difficile problema di ottimizzazione non lineare, in cui le difficoltà computazionali sono tipicamente dovute a

- **forti non linearità di E, che creano "valli ripide" e zone "piatte" nella superficie della funzione di errore;**
- **elevata dimensionalità di w ed elevato numero n_T di campioni;**
- **presenza di minimi locali non globali.**

Una ulteriore difficoltà è costituita dal fatto che gli insiemi di livello della funzione d'errore, ossia gli insiemi

$$\mathrm{L}(\alpha) = \{w \in R^m : E(w) \leq \alpha\}$$

possono non essere compatti, per cui non è possibile assicurare la convergenza globale di molti algoritmi a partire da punti iniziali arbitrari. E da notare, tuttavia, che in presenza di un termine di regolarizzazione *tutti* gli insiemi di livello sono compatti.

Uno dei primi algoritmi proposti per il calcolo dei pesi in una rete neurale è il metodo iterativo noto come ***metodo di backpropagation*** che è interpretabile come una versione euristica del *metodo del gradiente*. La riscoperta di questo metodo verso la metà degli anni '80 da parte di **Rumelhart , Hinton e Williams** ha reso possibile definire algoritmi di addestramento per reti multistrato e quindi è stata alla base del successivo considerevole sviluppo degli studi sulle reti neurali negli ultimi due decenni.

Sono state introdotte, in particolare, due classi principali di metodi iterativi per il calcolo dei pesi:

- **metodi *batch*** in cui ad ogni passo i pesi vengono aggiornati utilizzando informazioni relative a tutti i campioni dell'insieme di addestramento T;
- **metodi *on-line*** in cui ad ogni passo i pesi vengono aggiornati tenendo conto soltanto di un singolo campione di T.

I **metodi batch** possono essere utilizzati esclusivamente per l'addestramento *fuori linea*, supponendo di disporre di tutto l'insieme T prima di avviare il processo di minimizzazione.

I **metodi *on-line*** possono essere impiegati sia per l'addestramento fuori linea, sia per l'addestramento in tempo reale, cioè quando gli elementi di T vengono acquisiti durante il processo di addestramento.

I metodi batch sono ovviamente riconducibili a **metodi di ottimizzazione non vincolata per la minimizzazione di *C***. Infatti, nella letteratura neurale più recente sono sempre più frequentemente utilizzati, in luogo dei primi metodi euristici, metodi efficienti già sviluppati da tempo nel campo dell'ottimizzazione, che garantiscono la convergenza a *punti stazionari* della funzione di errore e usualmente assicurano una buona riduzione dell'obiettivo rispetto alla stima iniziale.

Il metodo di *backpropagation* (BP) è tuttora uno dei metodi di addestramento più diffusi. Il termine "**backpropagation**" (retropropagazione) è legato essenzialmente **alla tecnica utilizzata per il calcolo delle derivate della funzione di errore, basata sulle regole di derivazione delle funzioni composte**

Il metodo di BP è stato utilizzato in due versioni, note rispettivamente come:

- **BP *batch***, in cui i pesi vengono aggiornati dopo la presentazione di tutti i campioni del training set T;
- **BP *on-line*,** in cui i pesi vengono aggiornati in corrispondenza a ciascun campione di T.

La **BP batch** è definita dall'iterazione

$$w^{k+1} = w^k - \eta \nabla C(w^k),$$

dove $\nabla C(w^k)$ è il *gradiente* di C nel vettore corrente w^k e lo scalare $\eta > 0$ (detto *learning rate*) definisce il passo lungo l'antigradiente

La BP on-line consiste invece nel selezionare ad ogni passo un campione $(x^{j(k)}, y^{j(k)})$ dell'insieme di addestramento e nell'aggiornare i pesi utilizzando soltanto il termine $\nabla L_{j(k)}$ del gradiente di C, ossia nel porre:

$$w^{k+1} = w^k - \eta \nabla L_{j(k)}\left(w^k\right),$$

Il metodo di **BP è in genere inefficiente e può non convergere se il passo η non è scelto in modo appropriato;** nella letteratura neurale sono state quindi considerate **varie tecniche euristiche per effettuare una scelta adattativa del passo e per modificare la direzione di ricerca**.

Adesso introduciamo la procedura di calcolo del gradiente mediante backpropagation, che si può ricondurre a una particolare tecnica di *differenziazione automatica* .

Richiamiamo prima i seguenti strumenti di calcolo differenziale utili per la comprensione della backpropagation

Derivata di una funzione composta
Chain rule

- Se $g: R \to R$ e $f: R \to R$ sono derivabili, allora $g \circ f : R \to R$ è differenziabile,

  e se poniamo $\boldsymbol{h(x) = g(f(x))}$

- $\boldsymbol{\frac{dh}{dx} = g'(f(x)) \cdot f'(x)}$

- Se $q: R \to R$ e $g: R \to R$ e $f : R \to R$ sono derivabili, allora $q \circ g \circ f : R \to R$ è derivabile,

$$\boldsymbol{h(x) = q(g(f(x)))}$$

- $\boldsymbol{\frac{dh}{dx} = q'(g(f(x)) \cdot g'(f(x)) \cdot f'(x)}$

Derivata composta di funzioni di più variabili reali:

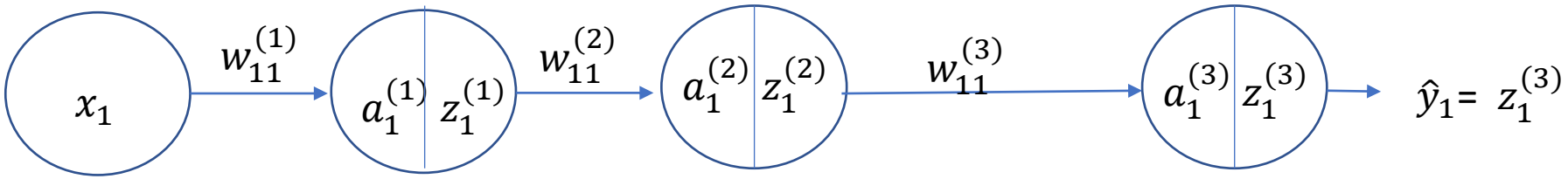
- Se $x(t) = \big(x_1(t), x_2(t), .., x_n(t)\big)$ è un vettore di R^n le cui componenti sono funzioni derivabili e se f è una funzione differenziabile in $x(t)$, allora la funzione composta $F(t) = f(x(t))$ è differenziabile nella variabile t e si ha:

$$F'(t) = \sum_{i=1}^{n} \frac{\partial f(x(t))}{\partial x_i} \cdot x_i'(t) =< \nabla f\big(x(t)\big), x'(t) >$$

- Esempio: data la funzione $f\big(h(t), g(t)\big)$, la derivata di f rispetto a t si calcola come:

$$\frac{df}{dt} = \frac{df}{dh}\frac{dh}{dt} + \frac{df}{dg}\frac{dg}{dt}$$

Consideriamo il seguente esempio di rete MLP molto semplice, formata da un nodo di input, 2 layer nascosti ciascuno dei quali costituito da un solo neurone ed un nodo di output:



Sia y_1 l'etichetta di x_1. La loss function che misura l'errore tra l'output prodotto dalla rete e il valore atteso è $L(y_1, \hat{y}_1)$.
Ricordiamo che il nostro compito è, in questo semplice esempio, aggiornare i pesi w in maniera tale da rendere minimo $L(y_1,\ \hat{y}_1)$, cioè individuare i pesi w tali che

$$\arg\min_{w} L(y_1,\ \hat{y}_1(w))$$

Per ottenere l'insieme di pesi $w = [w_{11}^{(1)}, w_{11}^{(2)}, w_{11}^{(3)}]$ ricorreremo al metodo di discesa del gradiente.

$$w^{k+1} = w^k - \eta \nabla L(w^k),$$

E' necessario calcolare:

$$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial w_{11}^{(1)}} \\ \frac{\partial L}{\partial w_{11}^{(2)}} \\ \frac{\partial L}{\partial w_{11}^{(3)}} \end{bmatrix}$$

$$L(\hat{y}_1) = L\left(z_1^{(3)}\right) = L\left(z_1^{(3)}(a_1^{(3)})\right) = L\left(z_1^{(3)}(a_1^{(3)}(z_1^{(2)})\right) = L\left(z_1^{(3)}(a_1^{(3)}(z_1^{(2)}(a_1^{(2)})\right) = L\left(z_1^{(3)}(a_1^{(3)}(z_1^{(2)}(a_1^{(2)}\left(z_1^{(1)}\right))\right) =$$

$$= L\left(z_1^{(3)}(a_1^{(3)}(z_1^{(2)}(a_1^{(2)}\left(z_1^{(1)}(a_1^{(1)})\right))\right) = L\left(z_1^{(3)}(a_1^{(3)}(z_1^{(2)}(a_1^{(2)}\left(z_1^{(1)}(a_1^{(1)}(w_{11}^{(1)})\right))\right)\right)$$

Calcoliamo la derivata parziale di L rispetto a $w_{11}^{(1)}$

$$\frac{\partial L}{\partial w_{11}^{(1)}} = \frac{\partial L}{\partial z_1^{(3)}} \frac{\partial z_1^{(3)}}{\partial a_1^{(3)}} \frac{\partial a_1^{(3)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial z_1^{(1)}} \frac{\partial z_1^{(1)}}{\partial a_1^{(1)}} \frac{\partial a_1^{(1)}}{\partial w_{11}^{(1)}}$$

$$\frac{\partial L}{\partial z_1^{(3)}} = \frac{\partial L}{\partial \hat{y}_1}$$

Si calcola facilmente derivando rispetto all'output della rete l'espressione analitica della loss-function

$$z_1^{(3)} = f(a_1^{(3)})$$

$$\frac{\partial z_1^{(3)}}{\partial a_1^{(3)}} = f'(a_1^{(3)})$$

$$a_1^{(3)} = w_{11}^{(3)} z_1^{(2)}$$

$$\frac{\partial a_1^{(3)}}{\partial z_1^{(2)}} = w_{11}^{(3)}$$

$$z_1^{(2)} = f(a_1^{(2)})$$

$$\frac{\partial z_1^{(2)}}{\partial a_1^{(2)}} = f'(a_1^{(2)})$$

$$a_1^{(2)} = w_{11}^{(2)} z_1^{(1)}$$

$$\frac{\partial a_1^{(2)}}{\partial z_1^{(1)}} = w_{11}^{(2)}$$

$$z_1^{(1)} = f(a_1^{(1)})$$

$$\frac{\partial z_1^{(1)}}{\partial a_1^{(1)}} = f'(a_1^{(1)})$$

$$a_1^{(1)} = w_{11}^{(1)} z_1^{(0)} \qquad z_1^{(0)} = x_1$$

$$\frac{\partial a_1^{(1)}}{\partial w_{11}^{(1)}} = z_1^{(0)}$$

$$\frac{\partial L}{\partial w_{11}^{(1)}} = \frac{\partial L}{\partial \hat{y}_1} f'(a_1^{(3)}) w_{11}^{(3)} f'(a_1^{(2)}) w_{11}^{(2)} f'(a_1^{(1)}) z_1^{(0)}$$

Calcoliamo la derivata parziale di L rispetto a $w_{11}^{(2)}$

$$L(\hat{y}_1) = L\left(z_1^{(3)}\right) = L\left(z_1^{(3)}(a_1^{(3)})\right) = L\left(z_1^{(3)}(a_1^{(3)}(z_1^{(2)})\right) = L\left(z_1^{(3)}(a_1^{(3)}(z_1^{(2)}(a_1^{(2)}(w_{11}^{(2)}))\right)$$

$$\frac{\partial L}{\partial w_{11}^{(2)}} = \frac{\partial L}{\partial z_1^{(3)}} \frac{\partial z_1^{(3)}}{\partial a_1^{(3)}} \frac{\partial a_1^{(3)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial w_{11}^{(2)}}$$

$$a_1^{(2)} = w_{11}^{(2)} z_1^{(1)}$$

$$\frac{\partial a_1^{(2)}}{\partial w_{11}^{(2)}} = z_1^{(1)}$$

$$\frac{\partial L}{\partial w_{11}^{(2)}} = \frac{\partial L}{\partial \hat{y}_1} \boldsymbol{f'(a_1^{(3)}) w_{11}^{(3)} f'\left(a_1^{(2)}\right)} z_1^{(1)}$$

Calcoliamo la derivata parziale di L rispetto a $w_{11}^{(3)}$

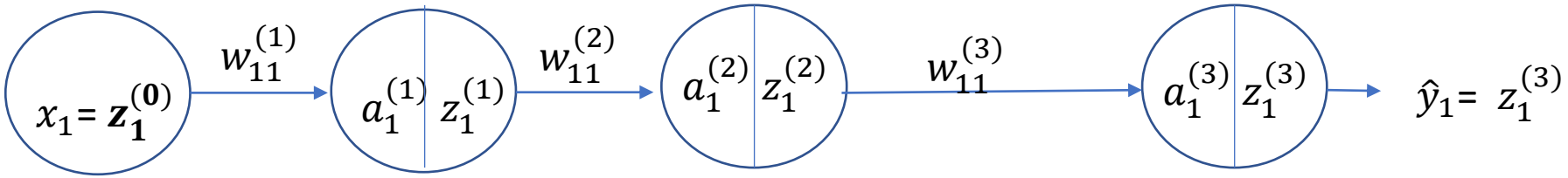
$$L(\hat{y}_1) = L\left(z_1^{(3)}\right) = L\left(z_1^{(3)}(a_1^{(3)})\right) = L\left(z_1^{(3)}(a_1^{(3)}(w_{11}^{(3)})\right)$$

$$\frac{\partial L}{\partial w_{11}^{(3)}} = \frac{\partial L}{\partial z_1^{(3)}} \frac{\partial z_1^{(3)}}{\partial a_1^{(3)}} \frac{\partial a_1^{(3)}}{\partial w_{11}^{(3)}}$$

$$a_1^{(3)} = w_{11}^{(3)} z_1^{(2)}$$

$$\frac{\partial a_1^{(3)}}{\partial w_{11}^{(3)}} = z_1^{(2)}$$

$$\frac{\partial L}{\partial w_{11}^{(3)}} = \frac{\partial L}{\partial \hat{y}_1} \boldsymbol{f'(a_1^{(3)})}\, z_1^{(2)}$$



$$\frac{\partial L}{\partial w_{11}^{(1)}} = \frac{\partial L}{\partial \hat{y}_1} f'(a_1^{(3)}) w_{11}^{(3)} f'\left(a_1^{(2)}\right) w_{11}^{(2)} f'\left(a_1^{(1)}\right) z_1^{(0)} =$$

$$\frac{\partial L}{\partial w_{11}^{(2)}} = \frac{\partial L}{\partial \hat{y}_1} f'(a_1^{(3)}) w_{11}^{(3)} f'\left(a_1^{(2)}\right) z_1^{(1)}$$

$$\frac{\partial L}{\partial w_{11}^{(3)}} = \frac{\partial L}{\partial \hat{y}_1} f'(a_1^{(3)}) z_1^{(2)}$$

Poniamo:

$$\delta_1^{(3)} = \frac{\partial L}{\partial \hat{y}_1} f'(a_1^{(3)})$$

$$\delta_1^{(2)} = \delta_1^{(3)} w_{11}^{(3)} f'\left(a_1^{(2)}\right)$$

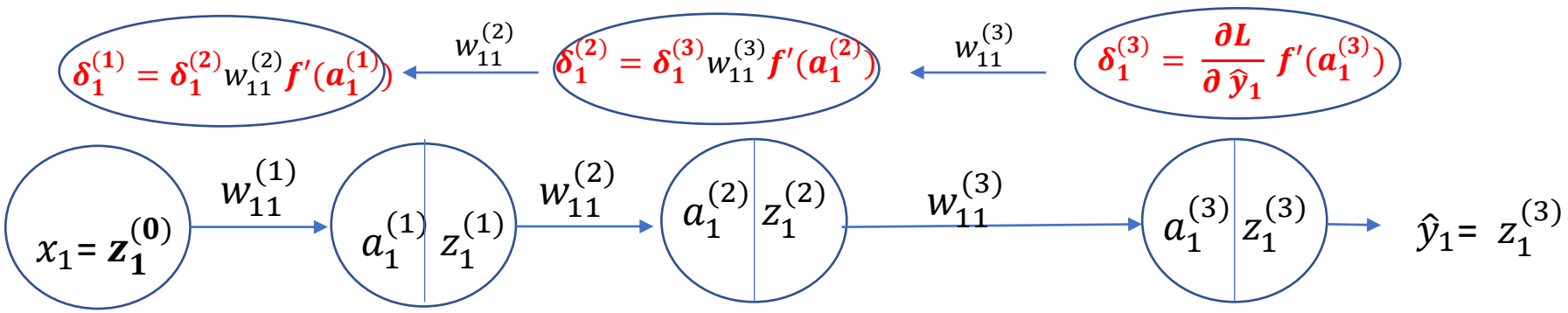
$$\delta_1^{(3)} = \delta_1^{(2)} w_{11}^{(2)} f'\left(a_1^{(1)}\right)$$

Le formule del gradiente della loss function L rispetto a tutti i pesi $\left\{w_{11}^{(1)}, w_{11}^{(2)}, w_{11}^{(3)}\right\}$si potranno quindi così esprimere:

$$\frac{\partial L}{\partial w_{11}^{(1)}} = \boldsymbol{\delta_1^{(1)} z_1^{(0)}}$$

$$\frac{\partial L}{\partial w_{11}^{(2)}} = \boldsymbol{\delta_1^{(2)} z_1^{(1)}}$$

$$\frac{\partial L}{\partial w_{11}^{(3)}} = \boldsymbol{\delta_1^{(3)} z_1^{(2)}}$$



Aggiornamento dei pesi per l'epoca successiva:

$$w^{k+1} = w^k - \eta \nabla L(w^k),$$

Omettendo per semplicità di scrittura l'indice k dell'epoca, l'ultima relazione è equivalente a:

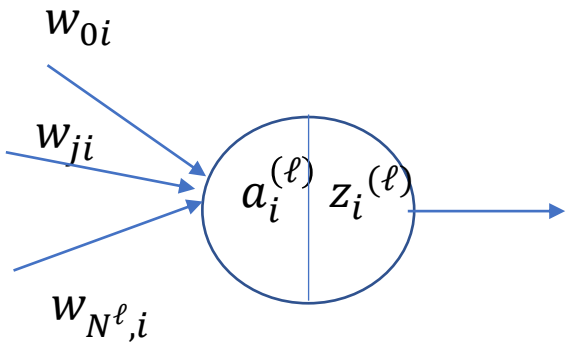
$$w_{11}^{(1)} = w_{11}^{(1)} - \eta \boldsymbol{\delta_1^{(1)}} x_1$$

$$w_{11}^{(2)} = w_{11}^{(2)} - \eta \boldsymbol{\delta_1^{(2)}} z_1^{(1)}$$

$$w_{11}^{(3)} = w_{11}^{(3)} - \eta \boldsymbol{\delta_1^{(3)}} z_1^{(2)}$$

Back propagation (dimostrazione)

$$a_i^{(\ell)} = \sum_{j=0}^{N^{(\ell)}} w_{ji}^{(\ell)} z_j^{(\ell-1)} \qquad z_i^{(\ell)} = f\left(a_i^{(\ell)}\right) \qquad z_i^{(0)} = x_i, \qquad i = 0, \dots, d$$



$$\frac{\partial L}{\partial w_{ji}^{(\ell)}} = \frac{\partial L}{\partial a_i^{(\ell)}} \frac{\partial a_i^{(\ell)}}{\partial w_{ji}^{(\ell)}}$$

$$a_i^{(\ell)} = \sum_{j=0}^{N^{(\ell)}} w_{ji}^{(\ell)} z_j^{(\ell-1)} = w_{0i}^{(\ell)} z_0^{(\ell-1)} + w_{1i}^{(\ell)} z_j^{(\ell-1)} + \cdots + w_{ji}^{(\ell)} z_j^{(\ell-1)} + \cdots w_{N^\ell i}^{(\ell)} z_{N^\ell}^{(\ell-1)}$$

$$\frac{\partial a_i^{(\ell)}}{\partial w_{ji}^{(\ell)}} = z_j^{(\ell-1)}$$

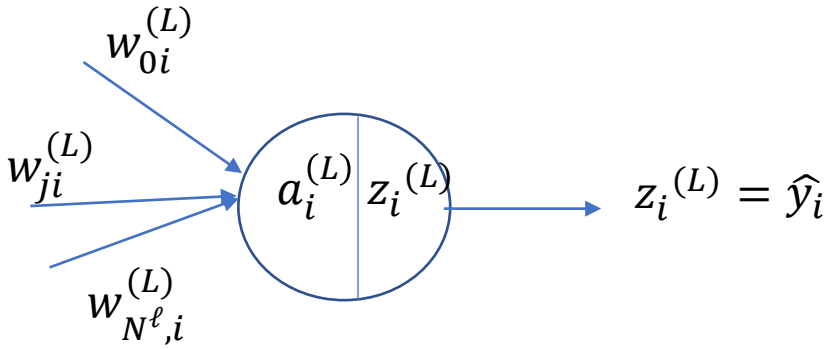
Definiamo $\delta_i^{(\ell)} := \frac{\partial L}{\partial a_i^{(\ell)}}$

$$\boldsymbol{\frac{\partial L}{\partial w_{ji}^{(\ell)}} = \delta_i^{(\ell)} z_j^{(\ell-1)}}$$

Per il calcolo di $\delta^{(i)}$ consideriamo due casi:

Caso 1: Il neurone appartiene allo **strato di uscita L**

$$\delta_i^{(L)} = \frac{\partial L(\hat{y}_i)}{\partial a_i^{(L)}}$$



$$L(\hat{y}_i) = L\left(z_i^{(L)}\right) = L\left(z_i^{(L)}(a_i^{(L)})\right)$$

$$\frac{\partial L(\hat{y}_i)}{\partial a_i^{(L)}} = \frac{\partial L(\hat{y}_i)}{\partial z_i^{(L)}} \frac{\partial z_i^{(L)}}{\partial a_i^{(L)}} = \frac{\partial L(\hat{y}_i)}{\partial \hat{y}_i} \frac{\partial z_i^{(L)}}{\partial a_i^{(L)}}$$

$\frac{\partial L(\hat{y}_i)}{\partial \hat{y}_i}$ viene calcolato analiticamente derivando la formula analitica della funzione loss.

Essendo

$$z_i^{(L)} = f\left(a_i^{(L)}\right)$$

segue:

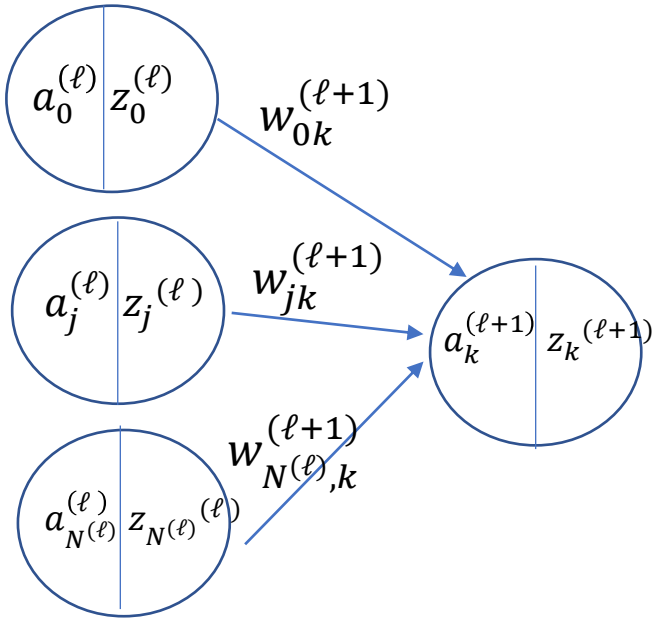
$$\frac{\partial z_i^{(L)}}{\partial a_i^{(L)}} = f'(a_i^{(L)})$$

Quindi, nel caso di **nodi appartenenti al layer di uscita si ha**:

$$\boldsymbol{\delta_i^{(L)} = \frac{\partial L(\hat{y}_i)}{\partial a_i^{(L)}} = f'\left(a_i^{(L)}\right)\frac{\partial L(\hat{y}_i)}{\partial \hat{y}_i}}$$

Caso 2: Il neurone **non** appartiene allo **strato di uscita**

Indichiamo **con *i* un neurone nascosto**:



Dobbiamo calcolare $\dfrac{\partial L}{\partial a_i^{(\ell)}}$

L dipende da $a_i^{(\ell)}$ tramite $a_k^{(\ell+1)}, k = 0, .., N^{(\ell)}$

Essendo $z_i^{(\ell)} = f\left(a_i^{(\ell)}\right)$

$$a_k^{(\ell+1)} = \sum_{j=0}^{N^{(\ell)}} w_{jk}^{(\ell+1)} z_j^{(\ell)} = \sum_{j=0}^{N^{(\ell)}} w_{jk}^{(\ell+1)} f(a_j^{(\ell)})$$

$$\delta_i^{(\ell)} := \frac{\partial L}{\partial a_i^{(\ell)}} = \sum_{k=0}^{N^{(\ell)}} \frac{\partial L}{\partial a_k^{(\ell+1)}} \frac{\partial a_k^{(\ell+1)}}{\partial a_i^{(\ell)}}$$

$$\delta_k^{(\ell+1)} := \frac{\partial L}{\partial a_k^{(\ell+1)}}$$

$$a_k^{(\ell+1)} = w_{0k}^{(\ell+1)} f(a_0^{(\ell)}) + w_{1k}^{(\ell+1)} f(a_1^{(\ell)}) + \ \ldots + w_{ik}^{(\ell+1)} f(a_i^{(\ell)}) + \cdots + w_{N^{(l)}k}^{(\ell+1)} f(a_{N^{(\ell)}}^{(\ell)})$$

Segue facilmente che

$$\frac{\partial a_k^{(\ell+1)}}{\partial a_i^{(\ell)}} = f'(a_i^{(\ell)}) w_{ik}^{(\ell+1)}$$

E quindi

$$\delta_i^{(\ell)} := \frac{\partial L}{\partial a_i^{(\ell)}} = \sum_{k=0}^{N^{(\ell)}} \delta_k^{(\ell+1)} f'(a_i^{(\ell)}) w_{ik}^{(\ell+1)}$$

che può essere riscritto come:

$$\boldsymbol{\delta_i^{(\ell)} = f'(a_i^{(\ell)}) \sum_{k=0}^{N^{(\ell)}} \delta_k^{(\ell+1)} w_{ik}^{(\ell+1)}}$$

Quindi,

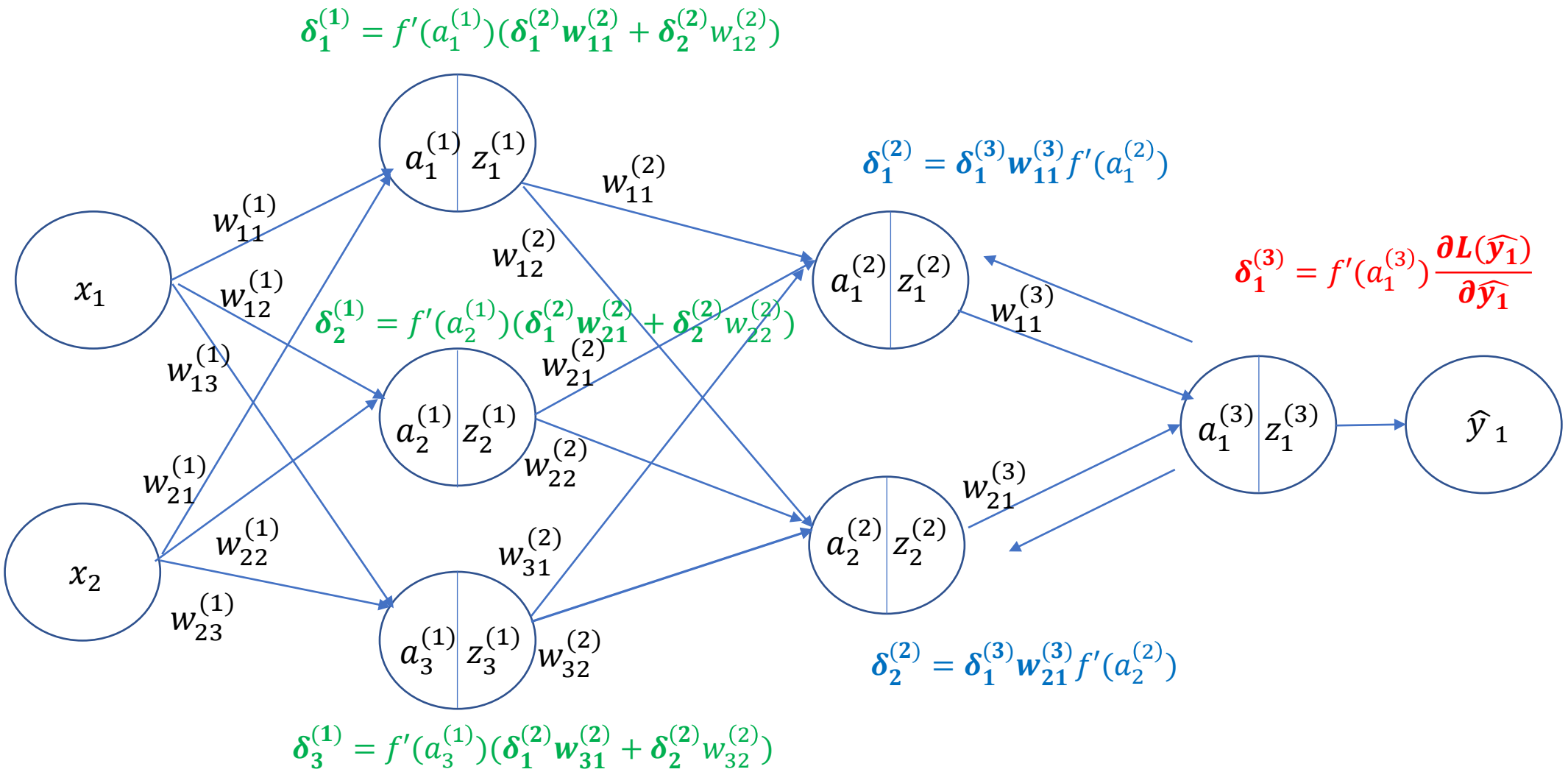
$$\frac{\partial L}{\partial w_{ji}^{(\ell)}} = \delta_i^{(\ell)} \, z_j^{(\ell-1)}$$

dove, se il neurone $i, i = 1, \dots, k$, appartiene al layer di uscita L,

$$\delta_i^{(L)} = \frac{\partial L(\hat{y}_i)}{\partial a_i^{(L)}} = f'\left(a_i^{(L)}\right) \frac{\partial L(\hat{y}_i)}{\partial \hat{y}_i}$$

se il neurone $i, i = 1, \dots, k$, non appartiene al layer di uscita, ma al layer nascosto ℓ,

$$\delta_i^{(\ell)} = f'(a_i^{(\ell)}) \sum_{k=0}^{N^{(\ell)}} \delta_k^{(\ell+1)} \, w_{ik}^{(\ell+1)}$$


$\boldsymbol{\delta_1^{(1)}} = f'(a_1^{(1)})(\boldsymbol{\delta_1^{(2)}} \boldsymbol{w_{11}^{(2)}} + \boldsymbol{\delta_2^{(2)}} w_{12}^{(2)})$
$\boldsymbol{\delta_1^{(2)}} = \boldsymbol{\delta_1^{(3)}} \boldsymbol{w_{11}^{(3)}} f'(a_1^{(2)})$
$\boldsymbol{\delta_1^{(3)}} = f'(a_1^{(3)}) \dfrac{\boldsymbol{\partial L(\widehat{y_1})}}{\boldsymbol{\partial \widehat{y_1}}}$
$\boldsymbol{\delta_2^{(1)}} = f'(a_2^{(1)})(\boldsymbol{\delta_1^{(2)}} \boldsymbol{w_{21}^{(2)}} + \boldsymbol{\delta_2^{(2)}} w_{22}^{(2)})$
$\boldsymbol{\delta_2^{(2)}} = \boldsymbol{\delta_1^{(3)}} \boldsymbol{w_{21}^{(3)}} f'(a_2^{(2)})$
$\boldsymbol{\delta_3^{(1)}} = f'(a_3^{(1)})(\boldsymbol{\delta_1^{(2)}} \boldsymbol{w_{31}^{(2)}} + \boldsymbol{\delta_2^{(2)}} w_{32}^{(2)})$
x_1
x_2
$a_1^{(1)}$ $z_1^{(1)}$
$a_2^{(1)}$ $z_2^{(1)}$
$a_3^{(1)}$ $z_3^{(1)}$
$a_1^{(2)}$ $z_1^{(2)}$
$a_2^{(2)}$ $z_2^{(2)}$
$a_1^{(3)}$ $z_1^{(3)}$
$\widehat{y}_1$
$w_{11}^{(1)}$
$w_{12}^{(1)}$
$w_{13}^{(1)}$
$w_{21}^{(1)}$
$w_{22}^{(1)}$
$w_{23}^{(1)}$
$w_{11}^{(2)}$
$w_{12}^{(2)}$
$w_{21}^{(2)}$
$w_{22}^{(2)}$
$w_{31}^{(2)}$
$w_{32}^{(2)}$
$w_{11}^{(3)}$
$w_{21}^{(3)}$

Sia $\hat{y} = [\hat{y}_1, .., \hat{y}_s]$ l'output prodotto dalla rete in corrispondenza all'esempio di training $x = [x_1, x_2, .. x_d]$, e sia $y = [y_1, y_2, .., y_s]$ il vettore delle etichette corrispondente dell'esempio di training.
Se scegliamo come **loss function** la somma dei quadrati degli errori tra l' output della rete e la corrispondente etichetta:

$$L(y, \hat{y}) = \frac{1}{2}\sum_{i=1}^{s}(y_i - \hat{y}_i)^2 = \frac{1}{2}\|y - \hat{y}\|_2^2$$

Ricordando che:

$$L(y, \hat{y}) = \frac{1}{2}((y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + \cdots (y_i - \hat{y}_i)^2 + \cdots + +(y_s - \hat{y}_s)^2$$

Segue facilmente che la derivata parziale di L rispetto all'output della rete $\hat{y}_i, i = 1, .., s$ sarà dato da :

$$\boldsymbol{\frac{\partial L}{\partial \hat{y}_i} = \frac{1}{2}\, 2(y_i - \hat{y}_i)(-1) = -(y_i - \hat{y}_i) = \hat{y}_i - y_i}$$