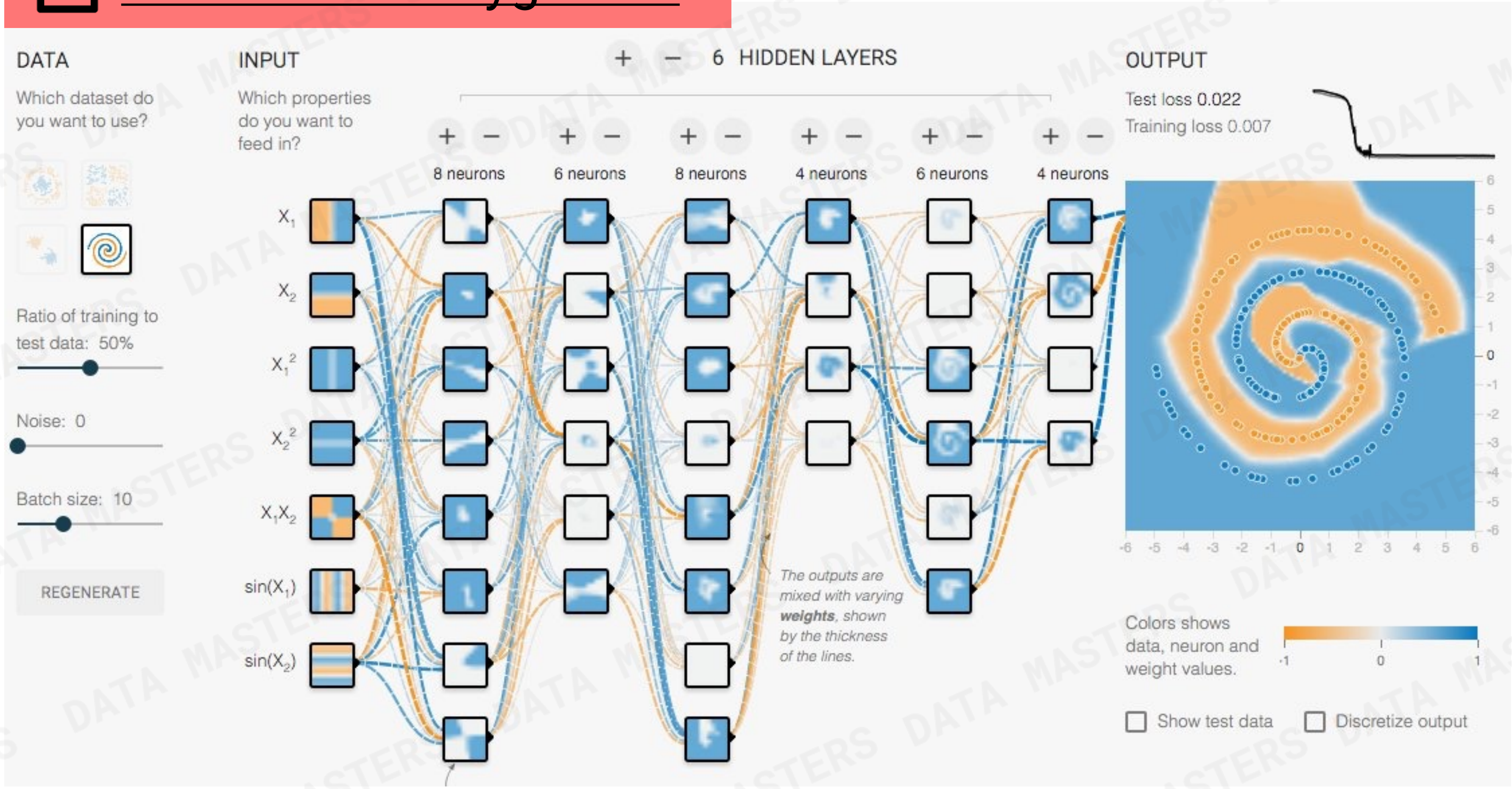
Esempi di reti neurali artificiali

[Tensorflow Playground]

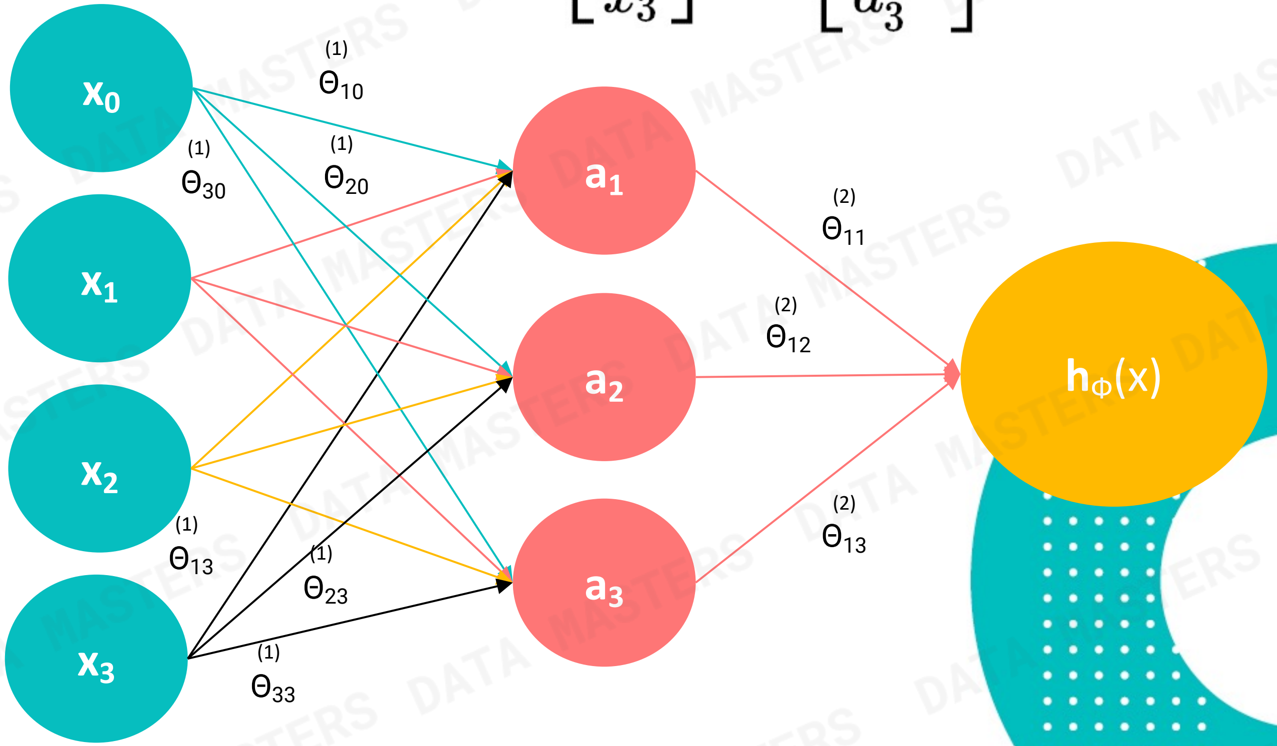


Rete Neurale Artificiale

$$\boldsymbol{h}_{\Theta}(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \rightarrow \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \end{bmatrix} \rightarrow h_\theta(x)$$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$



Ogni layer ha la sua matrice di pesi $\Phi^{(j)}$

La dimensione di $\Phi^{(j)}$ è data da:
(n° neuroni layer j+1) , (n° neuroni layer j)

Rete Neurale Artificiale – Funzione di Costo

Logistic regression:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Numero di classi di output

Neural network:

Numero di classi di output

$$h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{th} \text{ output}$$

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1-y_k^{(i)})\log(1-(h_\Theta(x^{(i)}))_k)\right]$$

Numero di layer

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

Numero di unità nel layer l

Rete Neurale Artificiale – Backpropagation

Lo **scopo** è **ottimizzare i parametri theta** che governano la funzione di costo, in modo da minimizzare quest'ultima:

$$\min_{\Theta} J(\Theta)$$

Errore del j-simo nodo nell'l-simo layer

Funzione di attivazione dell'ultimo layer (L=n°di layer)

- Per ogni nodo calcolare $\delta_j^{(l)}$

  che per l'ultimo layer è dato da $\delta^{(L)} = a^{(L)} - y$

  e per tutti gli altri si può applicare la formula:

$$\delta^{(l)} = \underbrace{((\Theta^{(l)})^T \delta^{(l+1)})}_{\text{Errore del nodo j,l-simo per il peso del nodo stesso}} \;.*\; \underbrace{\overbrace{a^{(l)} \;.*\; (1 - a^{(l)})}^{\text{Derivata prima della funzione sigmoide}}}_{\text{Direzione e verso dell'errore}}$$

Algoritmo di Backpropagation

Given training set $\{(x^{(1)}, y^{(1)}) \cdots (x^{(m)}, y^{(m)})\}$

- Set $\Delta_{i,j}^{(l)}$:= 0 for all (l,i,j)

Mantiene il «conto» dell'errore su ogni esempio, per tutti gli esempi del dataset

For training example t =1 to m:

- Set $a^{(1)} := x^{(t)}$
- Perform forward propagation to compute $a^{(l)}$ for l=2,3,…,L
- Using $y^{(t)}$, compute $\delta^{(L)} = a^{(L)} - y^{(t)}$
- Compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$ using $\delta^{(l)} = ((\Theta^{(l)})^T \delta^{(l+1)}) \mathbin{.*} a^{(l)} \mathbin{.*} (1 - a^{(l)})$
- $\Delta_{i,j}^{(l)} := \Delta_{i,j}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

$$\left.\begin{array}{l} D_{i,j}^{(l)} := \dfrac{1}{m}\left(\Delta_{i,j}^{(l)} + \lambda\Theta_{i,j}^{(l)}\right) \text{ If } j \neq 0 \\ D_{i,j}^{(l)} := \dfrac{1}{m}\Delta_{i,j}^{(l)} \text{ If } j=0 \end{array}\right\} = \frac{\partial}{\partial\Theta_{ij}^{(l)}} J(\Theta)$$

Esercitazione

Es04_artificial_neural_network.ipynb