DATA MASTERS

Dalle Regressioni alle Reti Neurali Artificiali

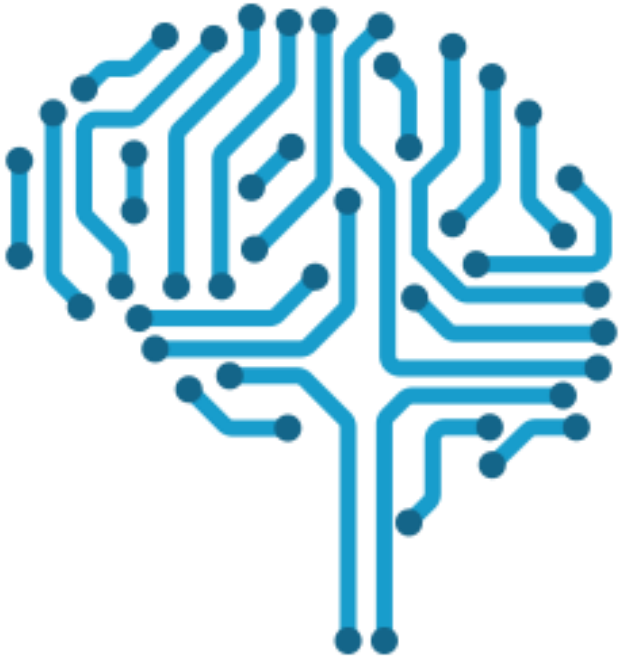
FrankHood

DATA MASTERS

Programma

- Regressione Lineare (*recap dal minicorso ML Starter Kit*)
- Regressione Logistica
- Funzione di costo per la Regressione Logistica
- Ottimizzazione tramite discesa del gradiente
- Regressione Lineare vs Logistica
- Regressione Logistica come neurone artificiale
- Reti Neurali Artificiali

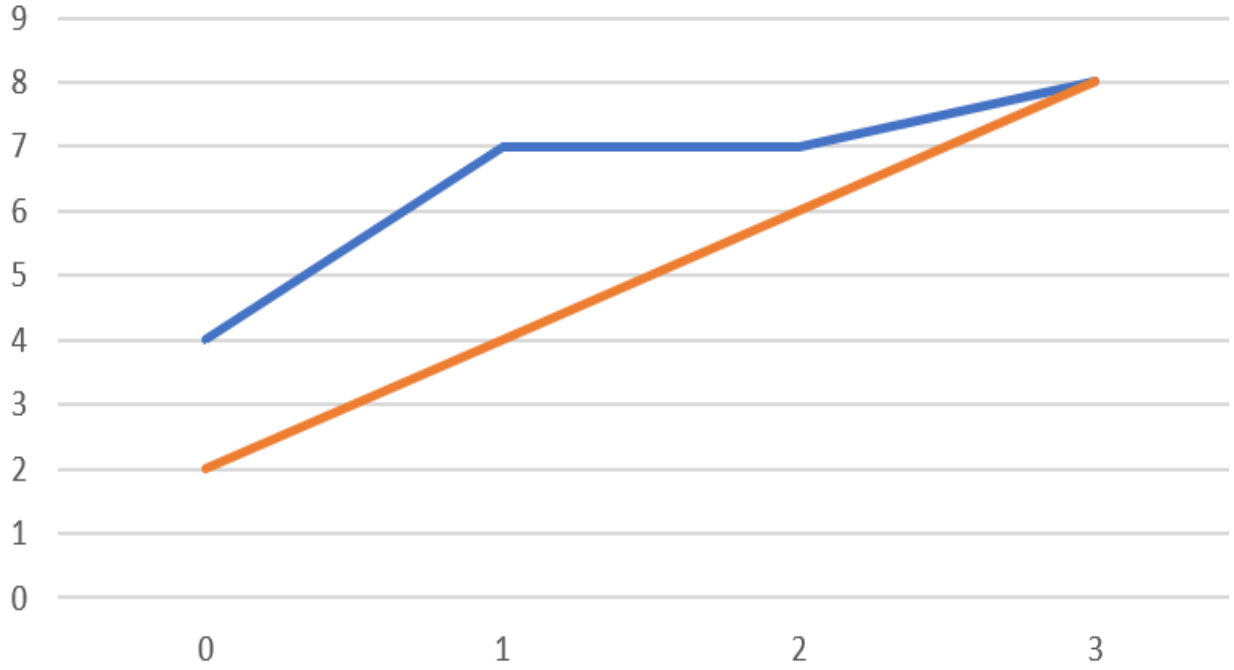
- Esercitazioni



DATA MASTERS

Esempio

INPUT (x)	OUTPUT (y)
0	4
1	7
2	7
3	8



$$\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x$$

Ipotesi: $\theta_0 = 2$ $\theta_1 = 2$ $h_\theta(x) = 2 + 2x$

Funzioni di costo: MSE

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

- Permette di misurare l'accuratezza del nostro modello
- Effettua una media tra i risultati delle ipotesi del modello rispetto ai valori riscontrati nella realtà
- E' chiamata Errore Quadratico Medio (*Mean Square Error - MSE*)

DATA MASTERS

Esempio di calcolo dell'MSE

x	y
0	4
1	7
2	7
3	8

$$h_\theta(x) = 2 + 2x$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

= (1/2*4) * [((2+2*0)-4)^2 + ((2+2*1)-7)^2 + ((2+2*2)-7)^2 + ((2+2*3)-8)^2]

= (1/8) * [(2-4)^2 + (4-7)^2 + (6-7)^2 + (8-8)^2]

= 0,125 * [4 + 9 + 1 + 0] = **1,75**

DATA MASTERS

Minizzare la funzione di costo

- Abbiamo una funzione di ipotesi
- Sappiamo misurare quanto tale funzione rappresenta dati reali

-> Dobbiamo stimare al meglio i parametri della funzione di ipotesi per minimizzare l'errore prodotto dalla nostra ipotesi rispetto ai casi reali

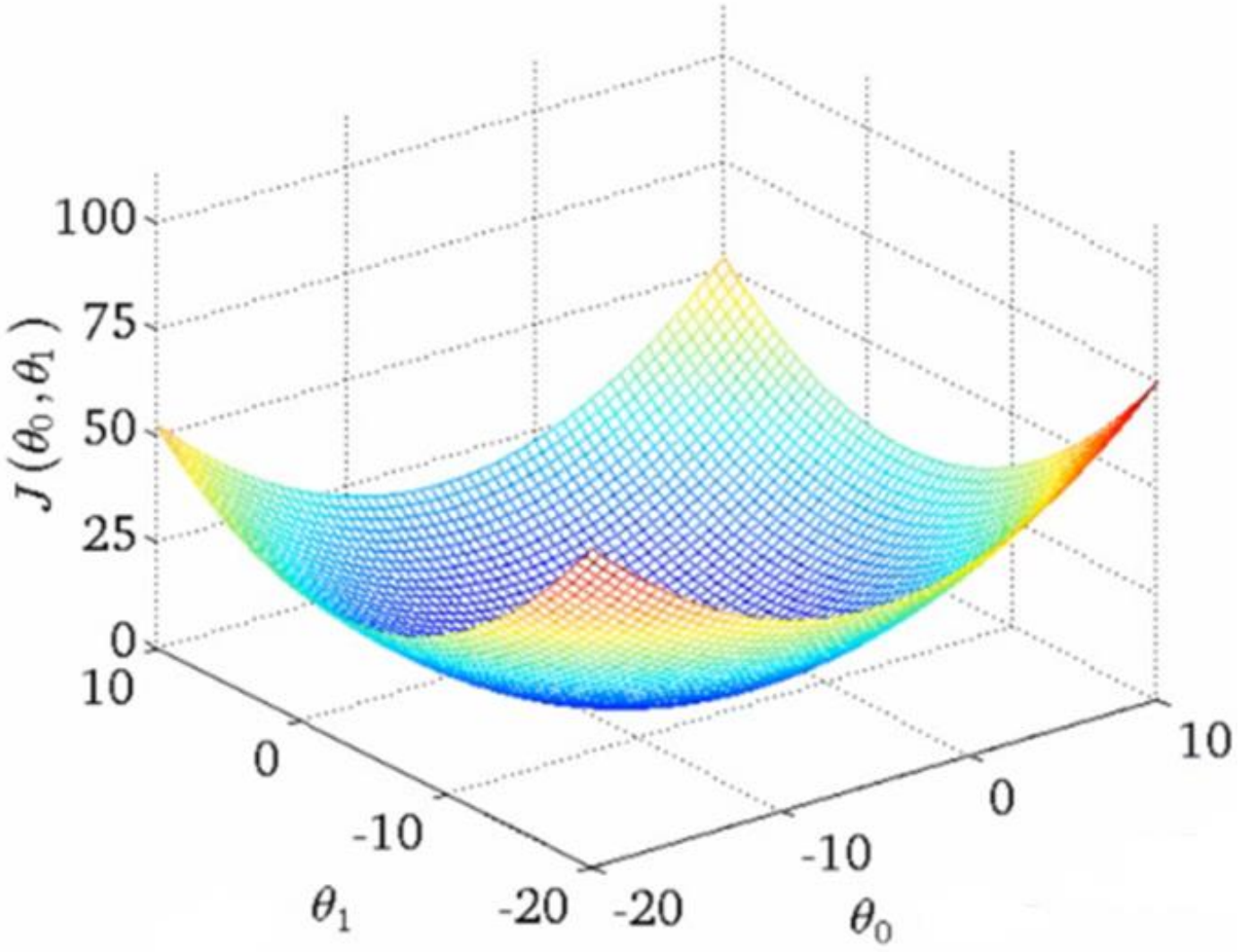
Come:

- In modo casuale
- Seguendo un metodo

DATA MASTERS

Discesa del Gradiente

Rappresentiamo graficamente la funzione di costo in base alla variazione dei parametri della funzione ipotesi

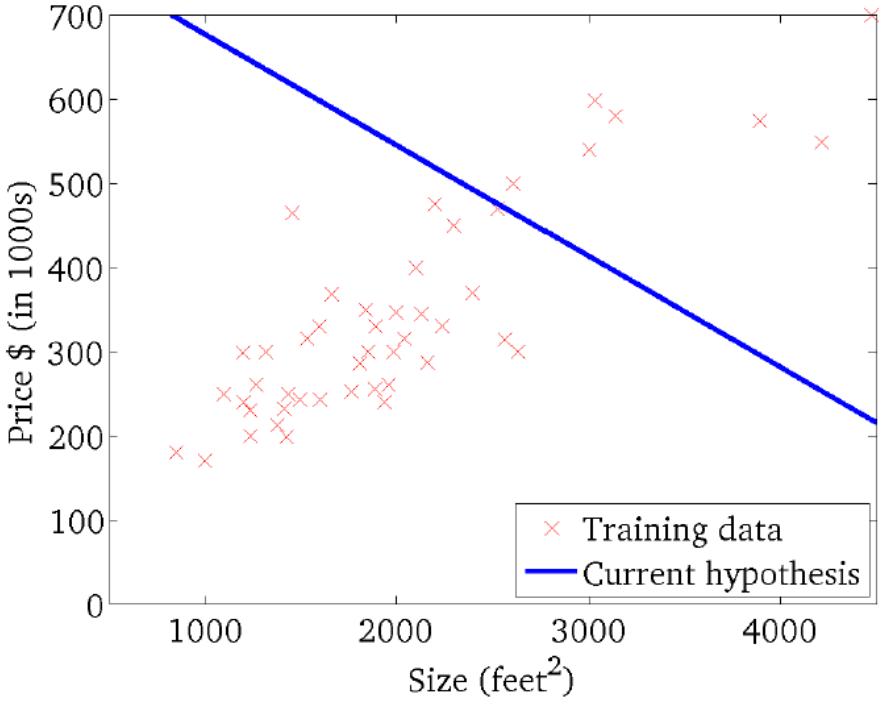


DATA MASTERS

Discesa del Gradiente (1/6)

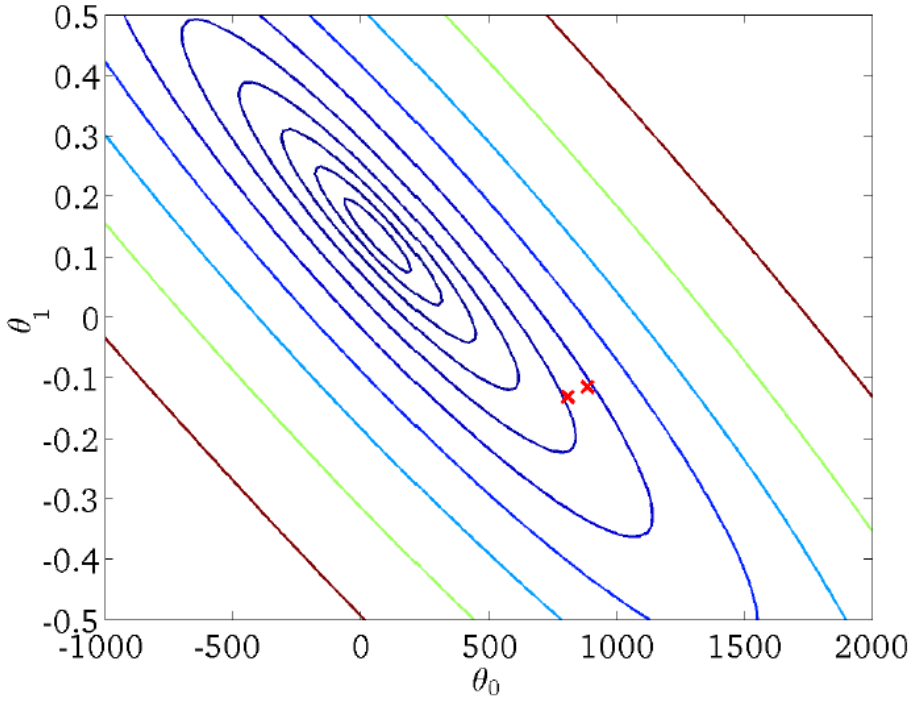
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

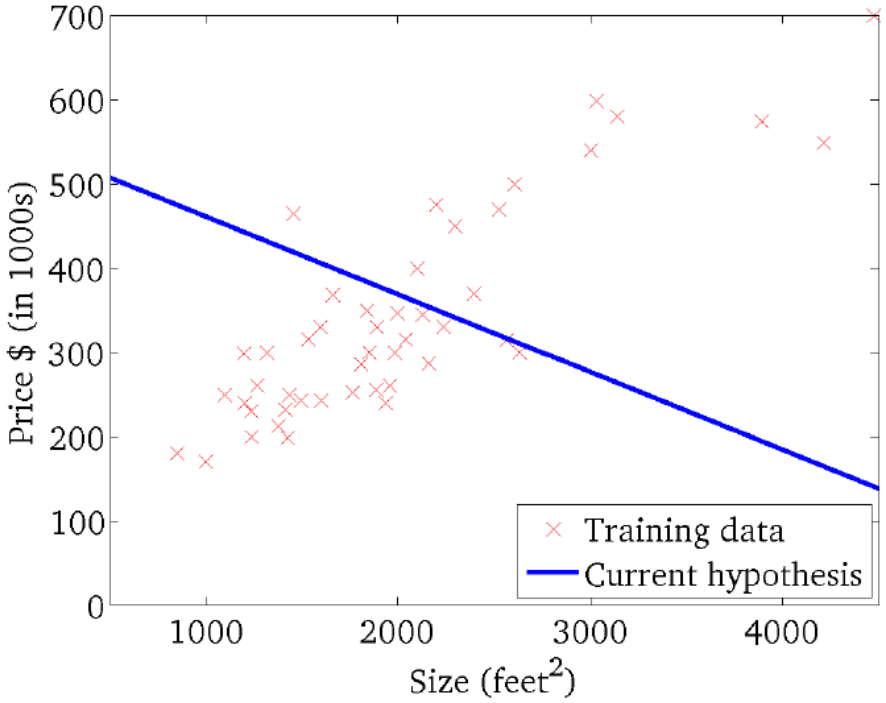


DATA MASTERS

Discesa del Gradiente (2/6)

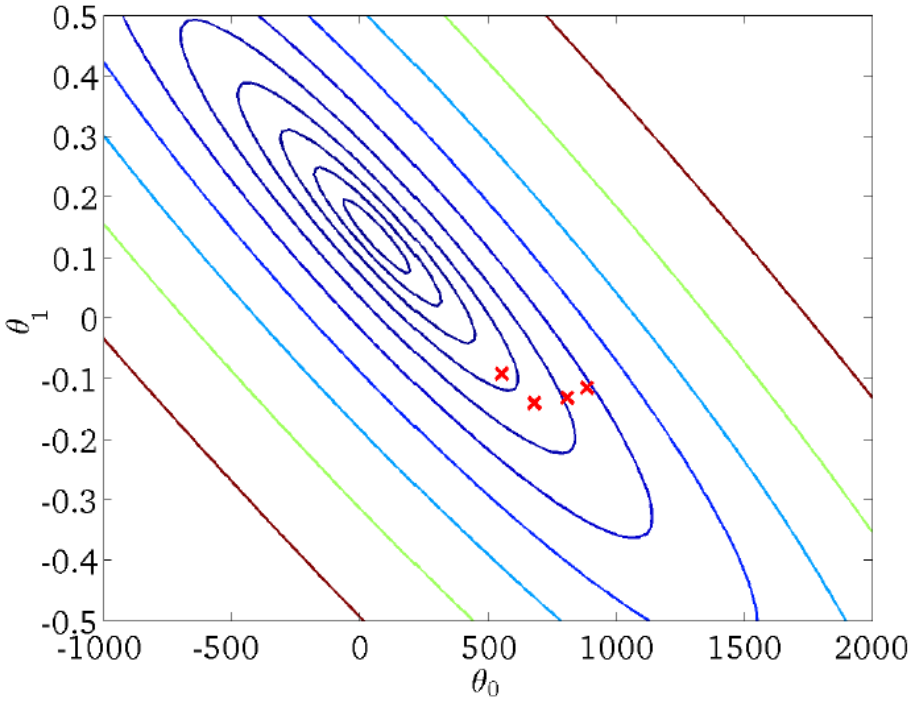
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

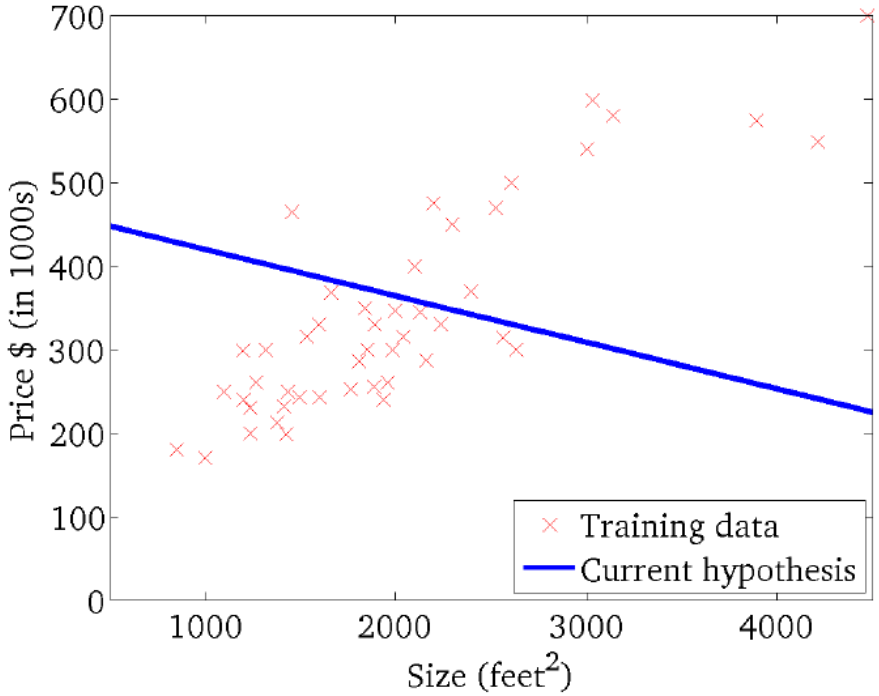


DATA MASTERS

Discesa del Gradiente (3/6)

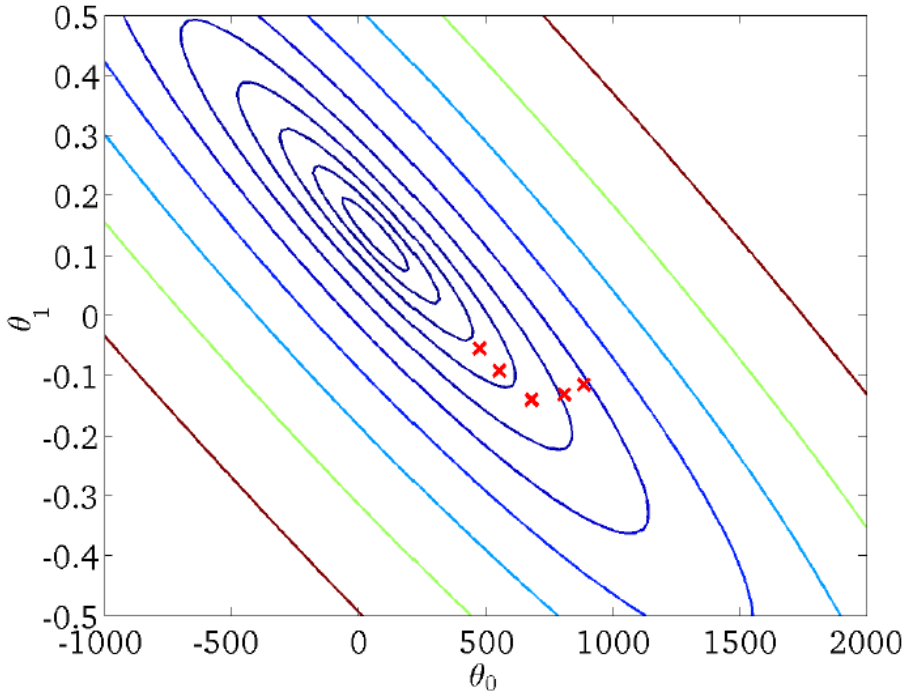
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

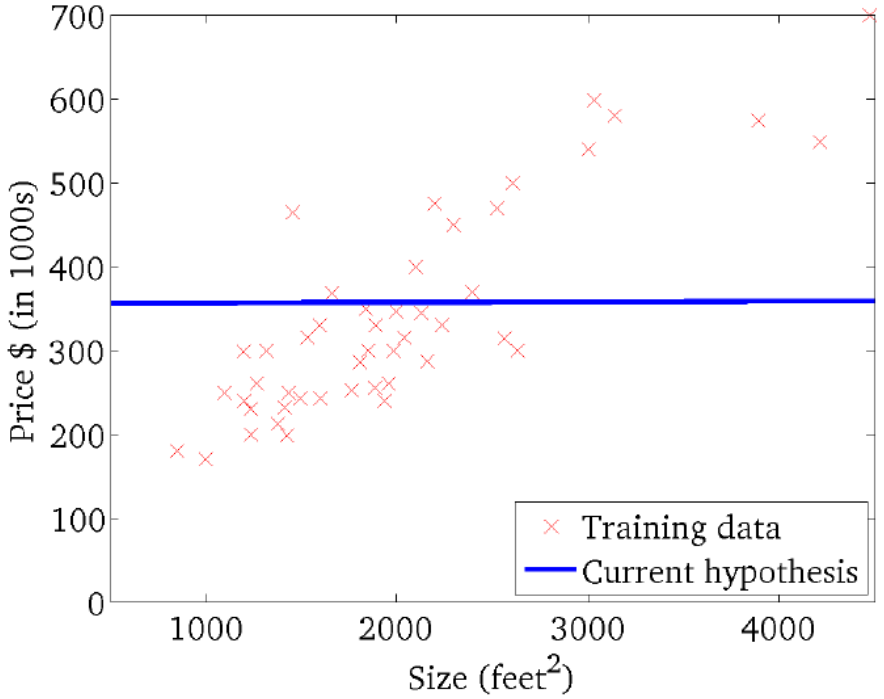


DATA MASTERS

Discesa del Gradiente (4/6)

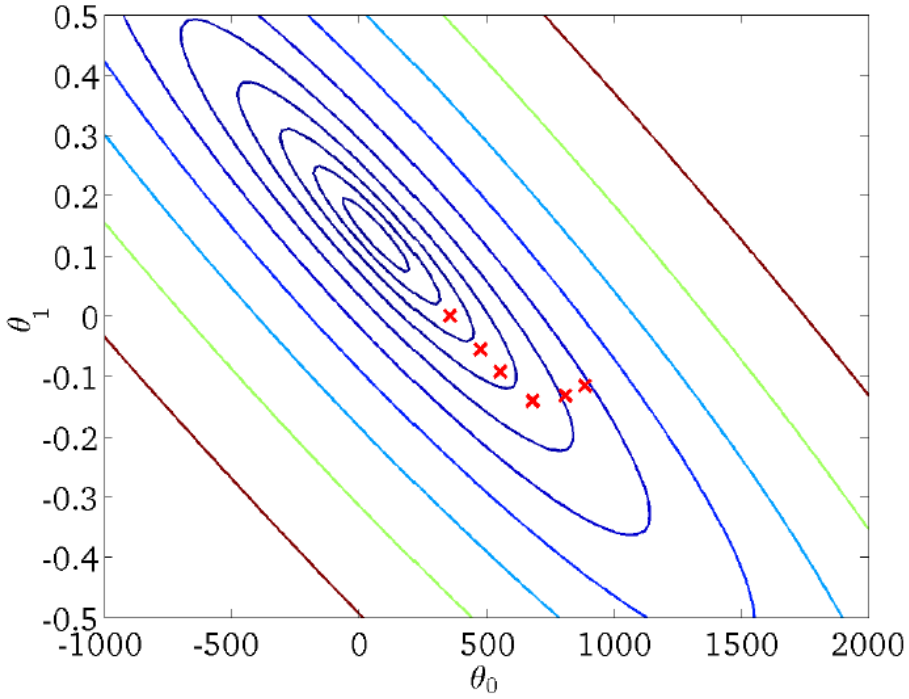
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

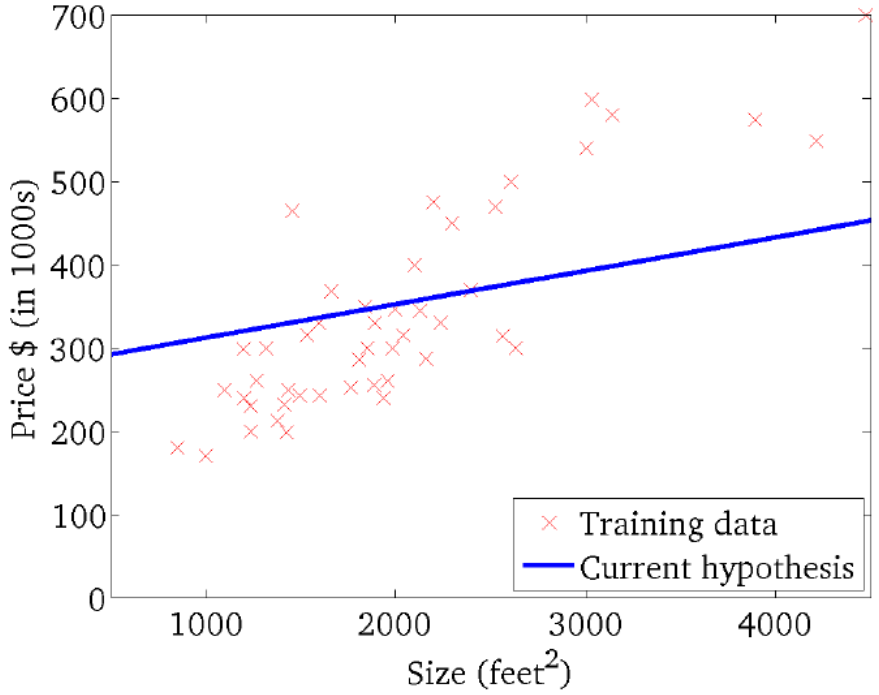


DATA MASTERS

Discesa del Gradiente (5/6)

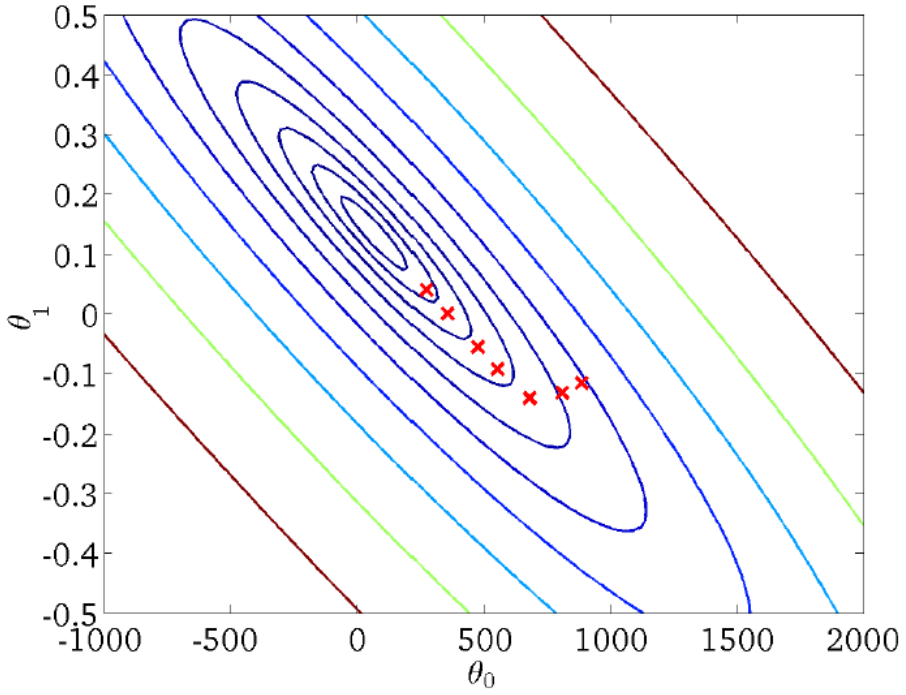
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

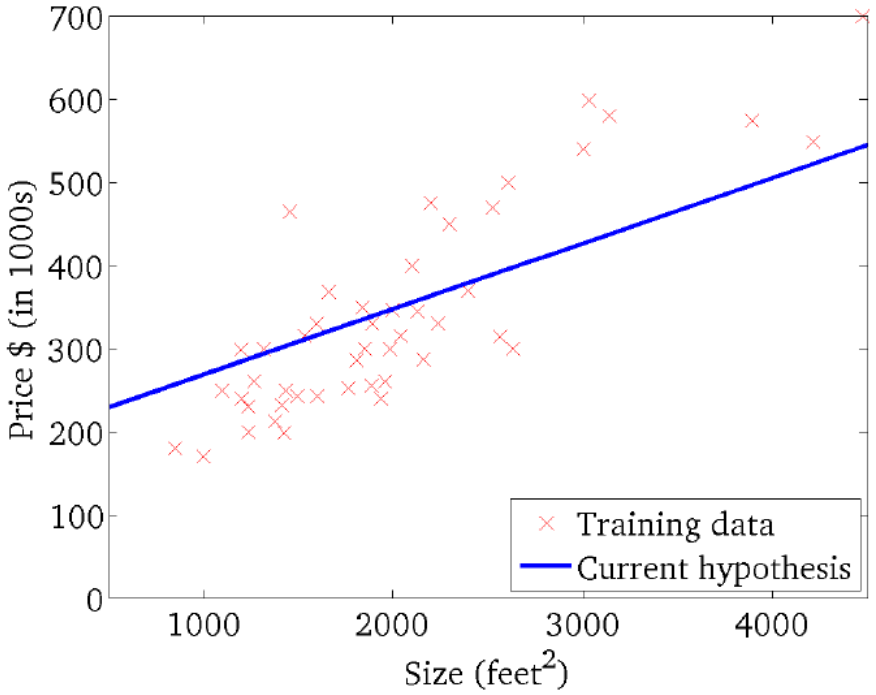


DATA MASTERS

Discesa del Gradiente (6/6)

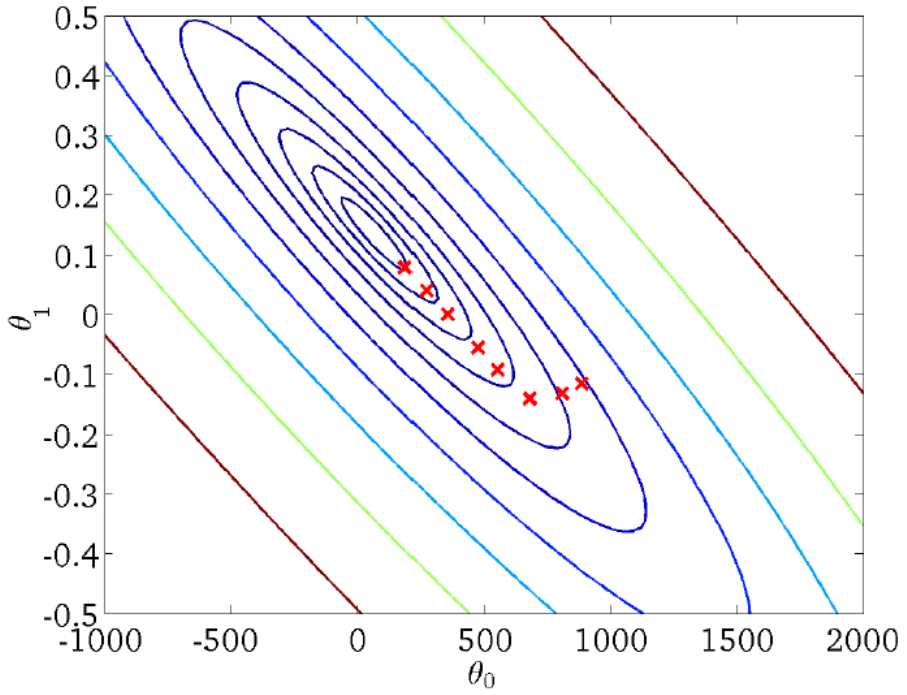
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



DATA MASTERS

Algoritmo della discesa del Gradiente

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

Learning Rate

repeat until convergence:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

repeat until convergence: {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x_i) - y_i) x_i)$$

}

DATA MASTERS

Algoritmo della discesa del gradiente

repeat until convergence: {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x_i) - y_i)x_i)$$

}

Si inizia impostando i parametri su valori casuali, quindi applicando ripetutamente la discesa del gradiente, la nostra ipotesi diventerà sempre più accurata.

Regressione Logistica

DATA MASTERS

Regressione Logistica

- E' un problema di classificazione
- L'output è composto da un numero finito di elementi

- **Classificazione binaria**: l'output può essere solo 0 o 1

- **Classificazione su classi multiple**: l'output è composto da più di due valori categorici

DATA MASTERS

Regressione Logistica

Usualmente si mappa su **0** un esito negativo e su **1** un esito positivo della osservazione

$$y \in \{0,1\}$$

	Two Class Classification	
$y \in \{0,1\}$	**1 or Positive Class**	**0 or Negative Class**
Email	Spam	Not Spam
Tumor	Malignant	Benign
Transaction	Fraudulent	Not Fraudulent

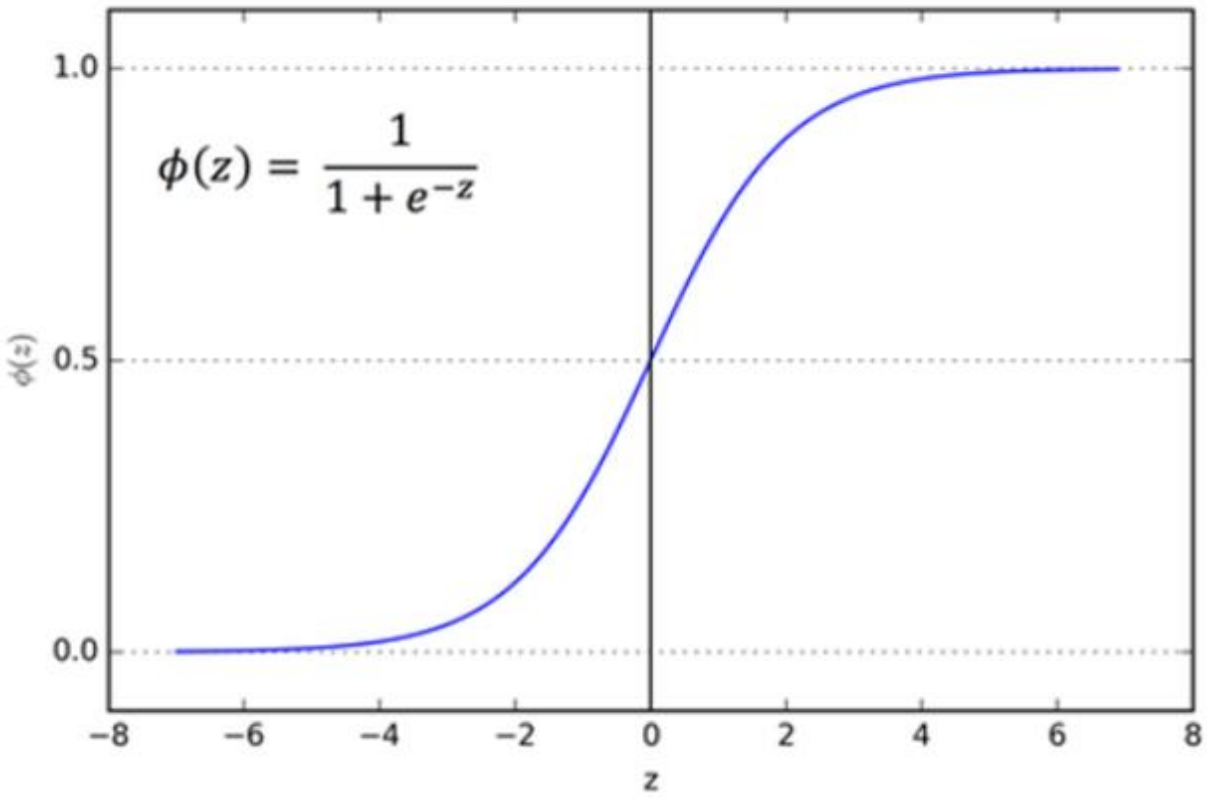
Classificazione Binaria

- La funzione ipotesi ha come risultato un numero tra 0 o 1
- Non è possibile utilizzare una funzione lineare perché, anche impostando un confine decisionale, i valori inferiori a zero o superiori ad 1 non avrebbero senso

DATA MASTERS

Funzione Sigmoide

Un esempio di funzione logistica è la funzione **sigmoide**, che restituisce un valore compreso tra 0 ed 1 per ogni valore reale passato in input



DATA MASTERS

Classificazione Binaria

Nel caso della regressione lineare si è utilizzata la formula:

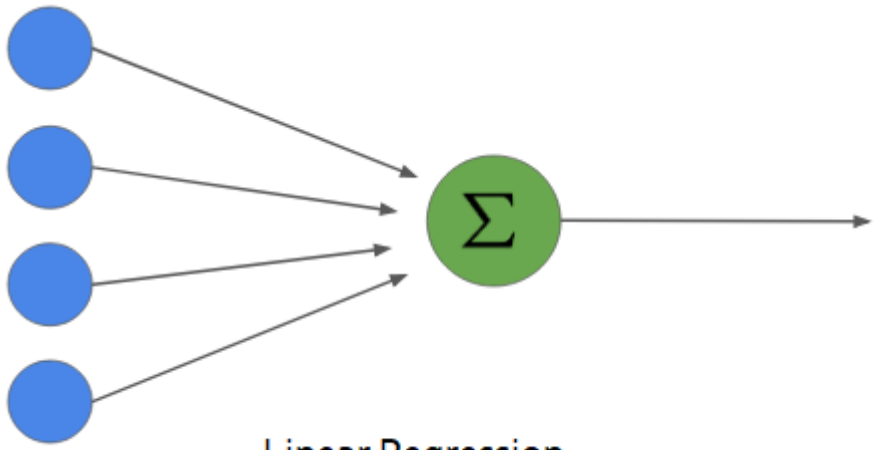
$$\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x$$

Per poterla sfruttare nel caso di una regressione logistica, la si può sfruttare componendola con la funzione sigmoide:

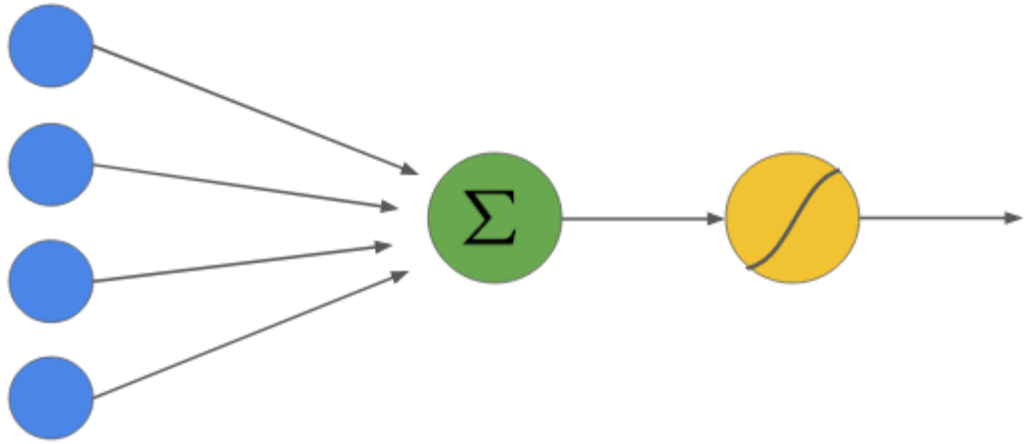
$$\hat{y} = h_\theta(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$$

DATA MASTERS

Regressione lineare vs logistica



Linear Regression



Logistic Regression

Classificazione Binaria

$h_\theta(x)$ fornisce la probabilità che l'output sia pari ad 1, ad esempio se risulta pari a 0.7 indica il 70% di probabilità che, per quel dato input, l'output sia 1

$$h_\theta(x) = P(y = 1|x;\theta) = 1 - P(y = 0|x;\theta)$$

$$P(y = 0|x;\theta) + P(y = 1|x;\theta) = 1$$

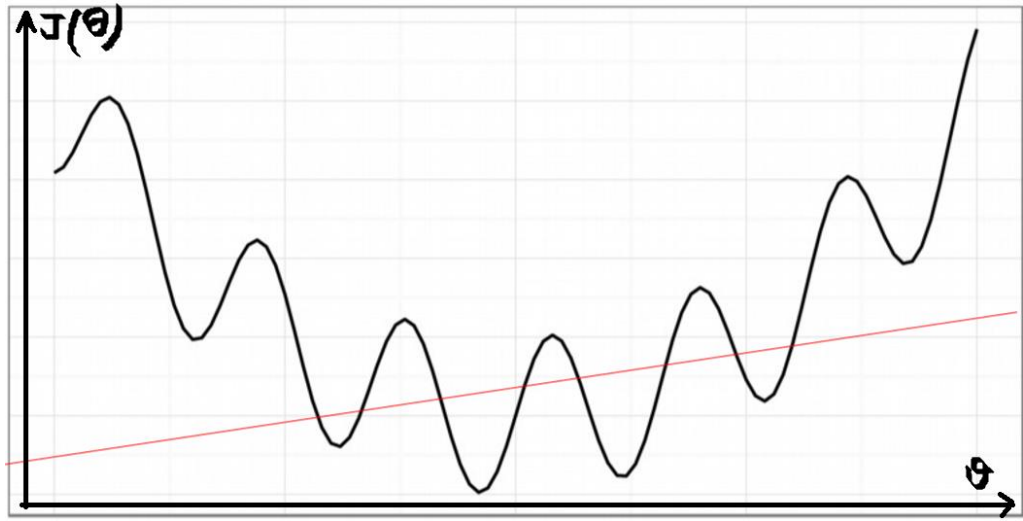
DATA MASTERS

Funzione di Costo per la Classificazione Binaria

Come nel caso della regressione lineare, rappresenta la funzione obiettivo dell'ottimizzazione del modello.

L'utilizzo della funzione di costo utilizzata nella regressione lineare risulta in una funzione non convessa, dove è più difficile trovare il minimo assoluto tra diversi minimi relativi

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$



DATA MASTERS

Funzione di Costo per la Classif.Binaria

Definizione di una funzione di costo «*comoda*» per la classificazione binaria, basandoci sui seguenti vincoli:

$$\text{Cost}(h_\theta(x), y) = 0 \text{ if } h_\theta(x) = y$$
$$\text{Cost}(h_\theta(x), y) \to \infty \text{ if } y = 0 \text{ and } h_\theta(x) \to 1$$
$$\text{Cost}(h_\theta(x), y) \to \infty \text{ if } y = 1 \text{ and } h_\theta(x) \to 0$$

$$\text{Cost}(h_\theta(x), y) = -y \log(h_\theta(x)) - (1 - y) \log(1 - h_\theta(x))$$

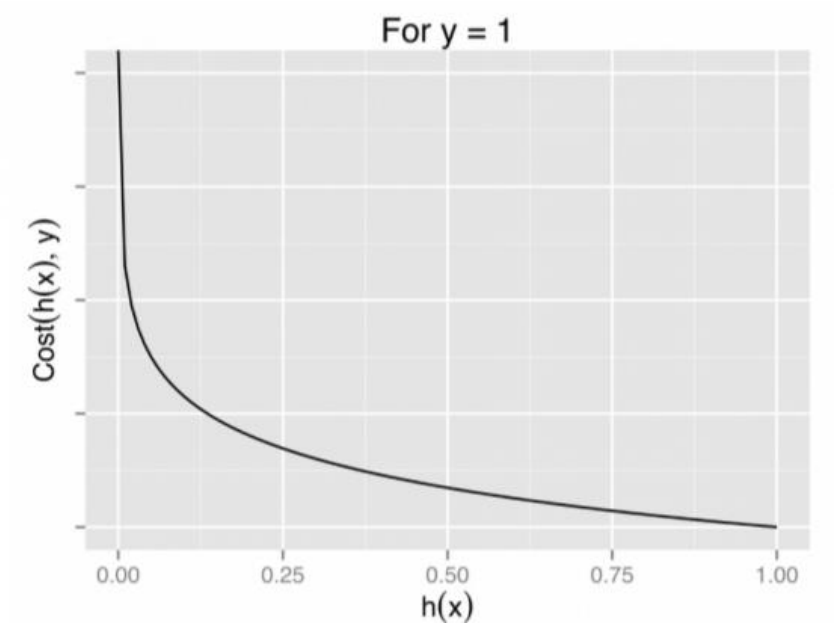
$$\text{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \qquad \text{if y} = 1$$
$$\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \qquad \text{if y} = 0$$

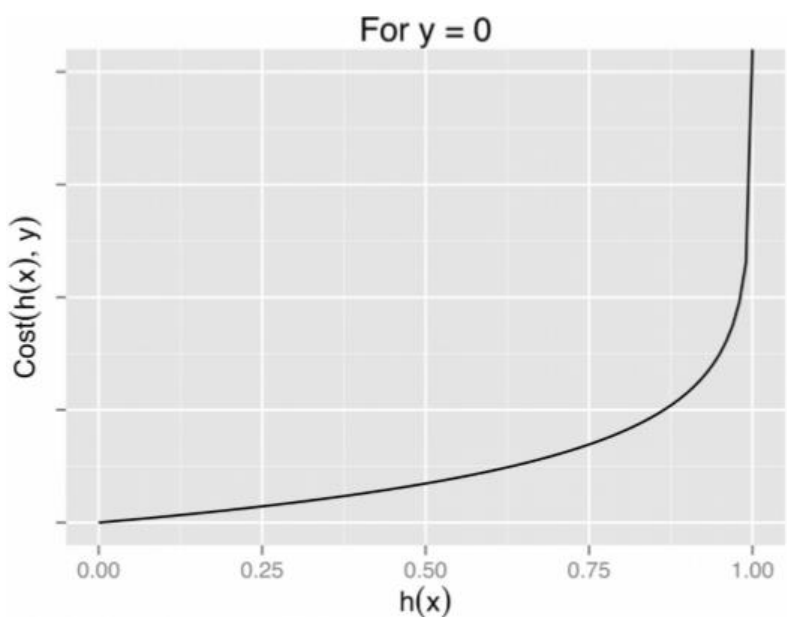
DATA MASTERS

Funzione di Costo per la Classif.Binaria

Analisi grafica della funzione di costo per la classificazione binaria:

$$\text{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \qquad \text{if y} = 1$$
$$\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \qquad \text{if y} = 0$$





Discesa del Gradiente per la Class.Binaria

Forma generale della discesa del gradiente:

repeat until convergence:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Ricordiamo la definizione della funzione di costo per la Classificazione Binaria:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$$

Derivata prima della funzione Sigmoide

$$\sigma(x)' = \sigma(x)(1 - \sigma(x))$$

$$\sigma(x)' = \left(\frac{1}{1+e^{-x}}\right)' = \frac{-(1+e^{-x})'}{(1+e^{-x})^2} = \frac{-1' - (e^{-x})'}{(1+e^{-x})^2} = \frac{0 - (-x)'(e^{-x})}{(1+e^{-x})^2} = \frac{-(-1)(e^{-x})}{(1+e^{-x})^2} = \frac{e^{-x}}{(1+e^{-x})^2}$$

$$= \left(\frac{1}{1+e^{-x}}\right)\left(\frac{e^{-x}}{1+e^{-x}}\right) = \sigma(x)\left(\frac{+1-1+e^{-x}}{1+e^{-x}}\right) = \sigma(x)\left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right) = \sigma(x)(1-\sigma(x))$$

DATA MASTERS

Derivate prime parziali della funzione di costo

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left[h_\theta(x^{(i)}) - y^{(i)} \right] x_j^{(i)}$$

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{-1}{m} \sum_{i=1}^{m} \left[y^{(i)} log(h_\theta(x^{(i)})) + (1 - y^{(i)}) log(1 - h_\theta(x^{(i)})) \right] \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \frac{\partial}{\partial \theta_j} log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \frac{\partial}{\partial \theta_j} log(1 - h_\theta(x^{(i)})) \right] \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[\frac{y^{(i)} \frac{\partial}{\partial \theta_j} h_\theta(x^{(i)})}{h_\theta(x^{(i)})} + \frac{(1 - y^{(i)}) \frac{\partial}{\partial \theta_j} (1 - h_\theta(x^{(i)}))}{1 - h_\theta(x^{(i)})} \right] \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[\frac{y^{(i)} \frac{\partial}{\partial \theta_j} \sigma(\theta^T x^{(i)})}{h_\theta(x^{(i)})} + \frac{(1 - y^{(i)}) \frac{\partial}{\partial \theta_j} (1 - \sigma(\theta^T x^{(i)}))}{1 - h_\theta(x^{(i)})} \right] \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[\frac{y^{(i)} \sigma(\theta^T x^{(i)})(1 - \sigma(\theta^T x^{(i)})) \frac{\partial}{\partial \theta_j} \theta^T x^{(i)}}{h_\theta(x^{(i)})} + \frac{-(1 - y^{(i)}) \sigma(\theta^T x^{(i)})(1 - \sigma(\theta^T x^{(i)})) \frac{\partial}{\partial \theta_j} \theta^T x^{(i)}}{1 - h_\theta(x^{(i)})} \right] \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[\frac{y^{(i)} h_\theta(x^{(i)})(1 - h_\theta(x^{(i)})) \frac{\partial}{\partial \theta_j} \theta^T x^{(i)}}{h_\theta(x^{(i)})} - \frac{(1 - y^{(i)}) h_\theta(x^{(i)})(1 - h_\theta(x^{(i)})) \frac{\partial}{\partial \theta_j} \theta^T x^{(i)}}{1 - h_\theta(x^{(i)})} \right] \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)}(1 - h_\theta(x^{(i)})) x_j^{(i)} - (1 - y^{(i)}) h_\theta(x^{(i)}) x_j^{(i)} \right] \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)}(1 - h_\theta(x^{(i)})) - (1 - y^{(i)}) h_\theta(x^{(i)}) \right] x_j^{(i)} \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} - y^{(i)} h_\theta(x^{(i)}) - h_\theta(x^{(i)}) + y^{(i)} h_\theta(x^{(i)}) \right] x_j^{(i)} \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} - h_\theta(x^{(i)}) \right] x_j^{(i)} \\
&= \frac{1}{m} \sum_{i=1}^{m} \left[h_\theta(x^{(i)}) - y^{(i)} \right] x_j^{(i)}
\end{aligned}$$

DATA MASTERS

Classificazione su più di due classi

$y = \{0,1...n\}$

In questo caso si divide il problema in n+1 (*+1 perché l'indice inizia da 0*) problemi di classificazione binaria; in ciascuno, si prevede la probabilità che 'y' sia un membro della relativa classe.

$$h_\theta^{(0)}(x) = P(y = 0|x;\theta)$$
$$h_\theta^{(1)}(x) = P(y = 1|x;\theta)$$
$$\dots$$
$$h_\theta^{(n)}(x) = P(y = n|x;\theta)$$

$$\text{prediction} = \max_i(h_\theta^{(i)}(x))$$

DATA MASTERS

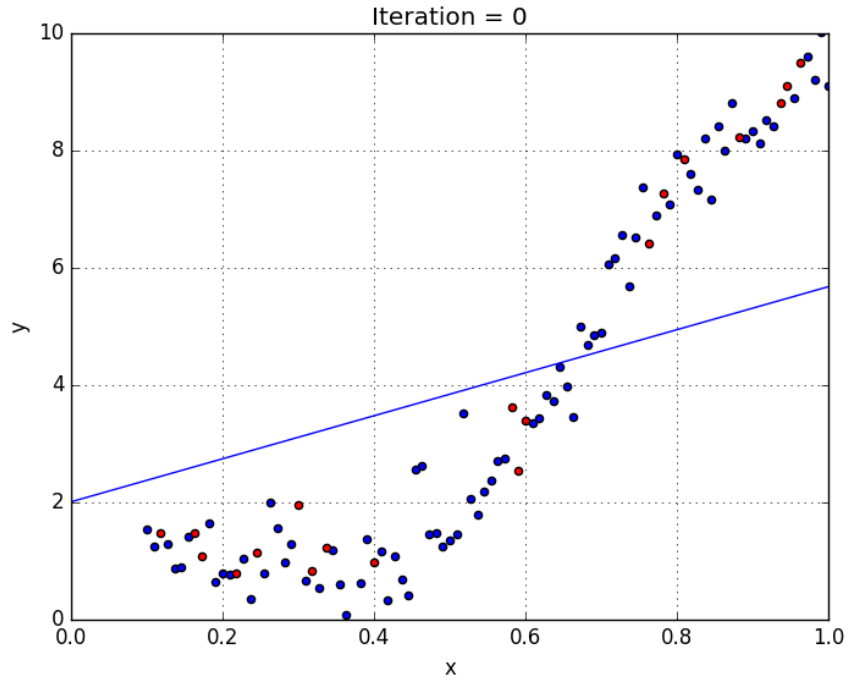
Regressione Logistica su più classi

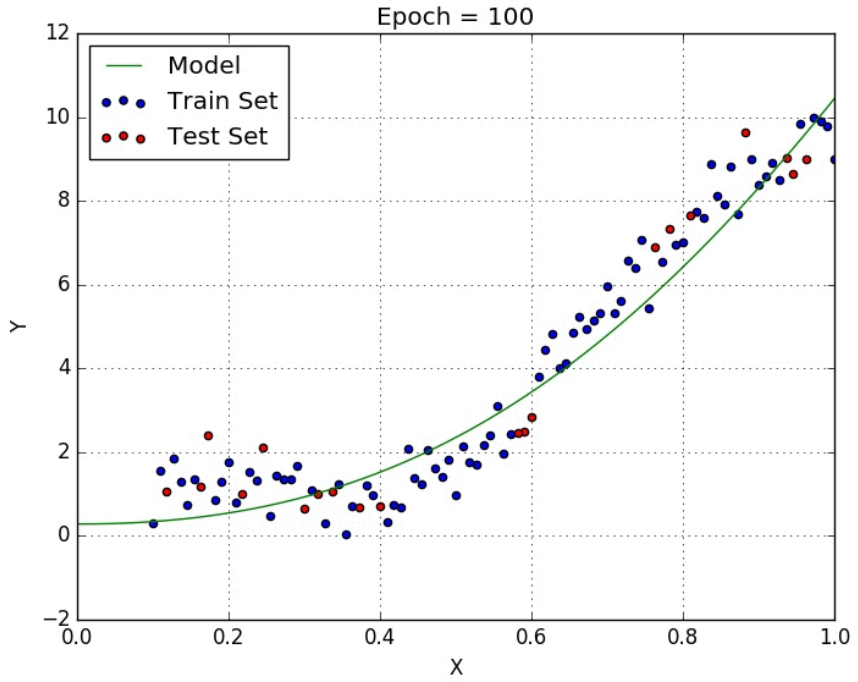
Esempio:

Es_1-regressione_logistica_su_più_classi.ipynb

DATA MASTERS

Regressione Lineare Polinomiale





$$y = \theta o + \theta_1 x_1 + \theta_2 x_1^2$$

DATA MASTERS

Regressione Polinomiale

- Vantaggi
  - riesce ad approssimare meglio la funzione che relaziona gli input agli output rispetto alla regressione lineare univariata
  - si adatta ad un numero maggiore di funzioni

- Svantaggi
  - la presenza di valori anomali inficia i risultati (outlier)
  - richiede potenza di calcolo maggiore

Reti Neurali Artificiali

DATA MASTERS

Ipotesi non lineari

Utilizzare la regressione su un set di features ampio è un compito arduo.
Se ad esempio volessimo creare un'ipotesi basandoci su tre features, includendo tutti i termini quadrati avremo 6 parametri:

$$g(\Phi_0+\Phi_1 x_1^2+\Phi_2 x_1 x_2+\Phi_3 x_1 x_3+\Phi_4 x_2^2+\Phi_5 x_2 x_3+\Phi_6 x_3^2)$$

Formula per calcolare il numero di parametri necessari:

$$\frac{(n+r-1)!}{r!(n-1)!}$$ con **n** = numero di features e **r** = esponente

(ad es. con 100 features avremo 5050 parametri da considerare)

DATA MASTERS

Ipotesi non lineari

Si può stimare il numero di parametri necessari approssimandolo a

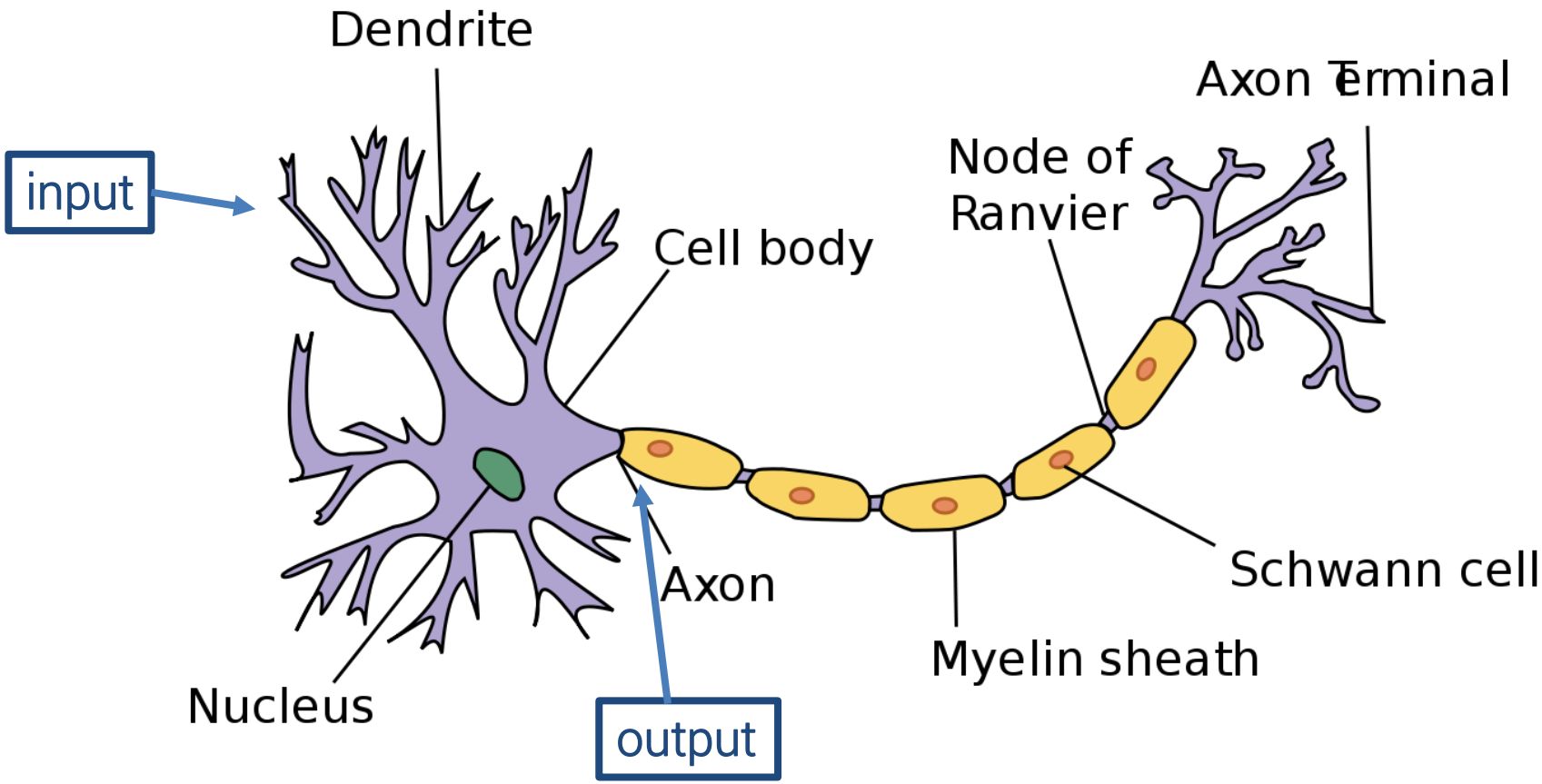
$n^2/2$

Considerando un dataset composto da foto da 50x50 pixels, avremo 2500 features; volendolo analizzare tramite regressione con termini quadratici avremo circa $2500^2 / 2 = 3.125.000$ parametri

» Le reti neurali artificiali offrono una strada alternativa per sfruttare il ML nei casi di ipotesi complesse e di un gran numero di features.

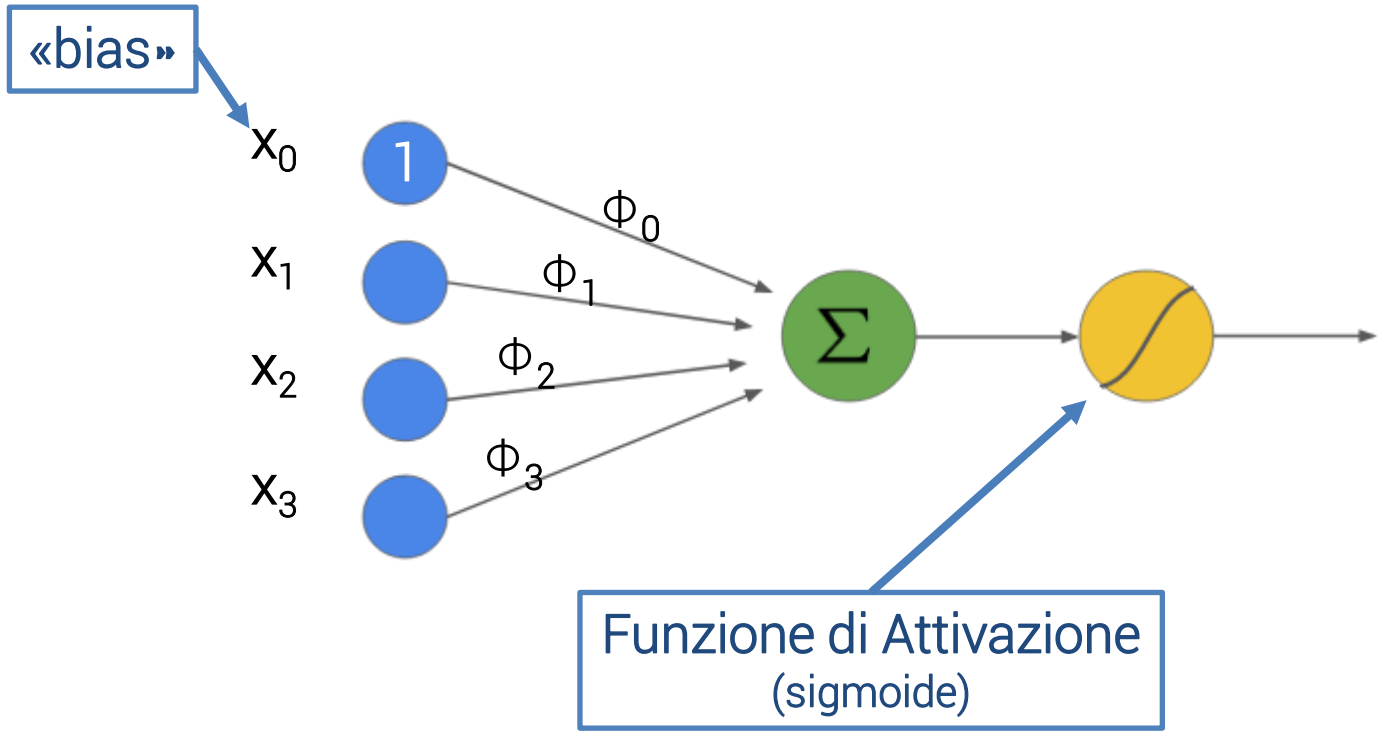
DATA MASTERS

Neurone umano



DATA MASTERS

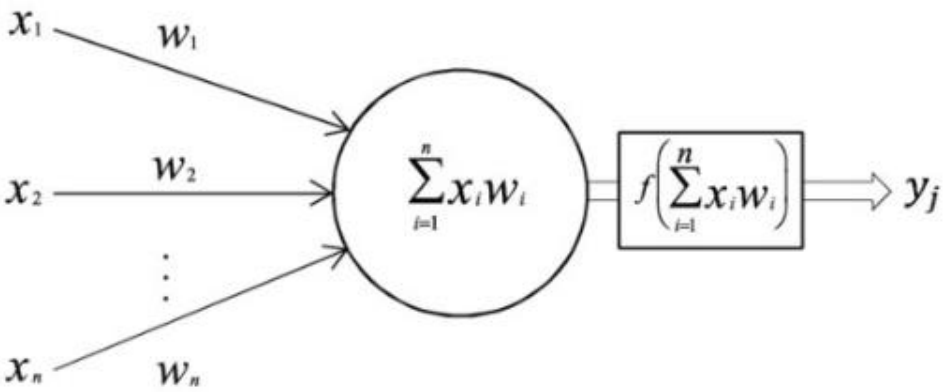
Regressione Logistica come Neurone Artificiale

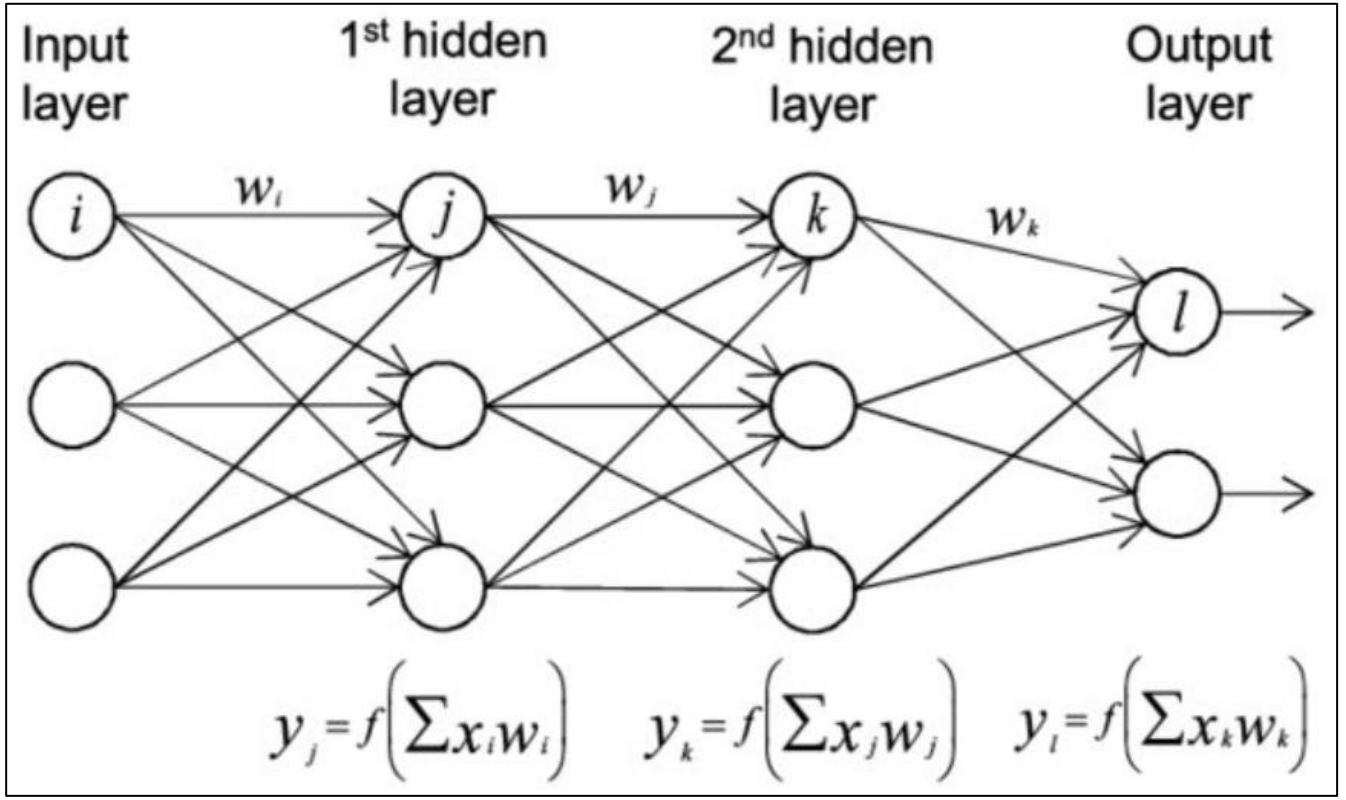


$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \Phi = \begin{bmatrix} \Phi_0 \\ \Phi_1 \\ \Phi_2 \\ \Phi_3 \end{bmatrix}$$

DATA MASTERS

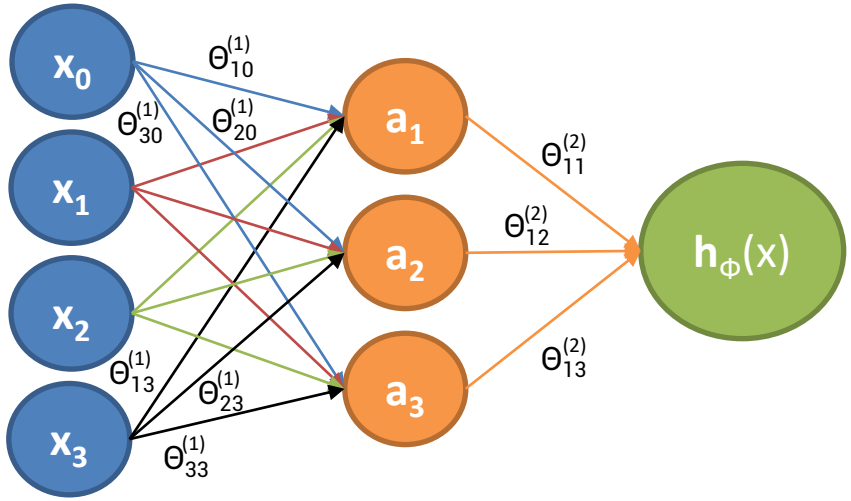
Rete Neurale Artificiale





DATA MASTERS

Rete Neurale Artificiale



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

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

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

DATA MASTERS

Funzione di Costo

Regressione Logistica

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

Regolarizzazione

Numero di classi di output

Numero di classi di input

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

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

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+1

Numero di unità nel layer l

DATA MASTERS

Backpropagation

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

$$\min_\Theta J(\Theta)$$

- Per ogni cella 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:

Errore della j-sima cella nell'l-simo layer

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

Derivata prima della funzione sigmoide

$$\delta^{(l)} = ((\Theta^{(l)})^T \delta^{(l+1)}) .* a^{(l)} .* (1 - a^{(l)})$$

Errore del nodo j,l-simo per il peso del nodo stesso

Direzione e verso dell'errore

DATA MASTERS

Esempi di Reti Neurali Artificiali

Esempio:

Tensorflow Playground

- Features lineari
- Features non lineari
- Features complesse