PRESENTA

Hello Machine Learning
dal punto di vista di un programmatore

Giuseppe Toto

OBIETTIVO DI QUESTO 2°GIORNO

- La **magia** dietro al ML
  - Partiamo da **zero**

The Pragmatic Programmers

Programming Machine Learning

From Coding to Deep Learning

Paolo Perrotta
edited by Katharine Dvorak

I compiti che può risolvere il MACHINE LEARNING?

I COMPITI CHE PUO' RISOLVERE IL ML

Scrivete il vostro

Codice di Avviamento Postale

(CAP)

I COMPITI CHE PUO' RISOLVERE IL ML

Costruiamo un

sistema di scansione

CAP

I COMPITI CHE PUO' RISOLVERE IL ML

Sono un programmatore:

«Come lo traduco in un algoritmo?»

I COMPITI CHE PUO' RISOLVERE IL ML

Sono un programmatore:

~~«Come lo traduco in un algoritmo?»~~

MACHINE LEARNING

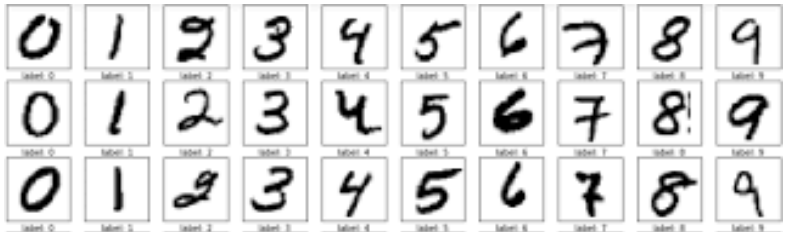
Addestriamo il nostro sistema

MACHINE LEARNING

COSTRUISCO IL TRAINING SET

MACHINE LEARNING:
sistema di apprendimento supervisionato

TRAINING SET

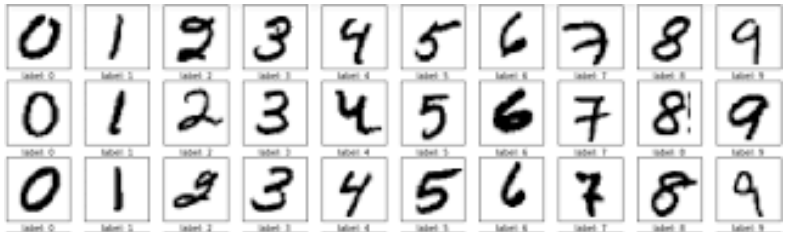


esempi etichettati

COSTRUISCO IL TEST SET

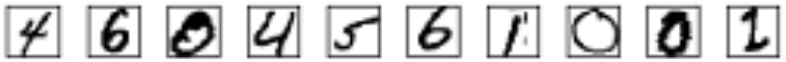
MACHINE LEARNING:
sistema di apprendimento supervisionato

TRAINING SET



esempi etichettati

TEST SET



MACHINE LEARNING

Alberi di decisione

Regressione lineare

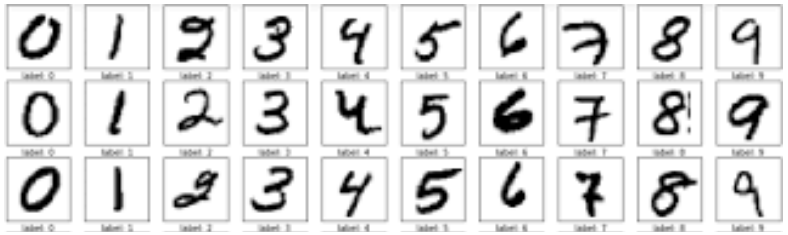
SCELGO UN MODELLO MATEMATICO

Reti neurali

MACHINE LEARNING:
sistema di apprendimento supervisionato

scelgo un
MODELLO MATEMATICO

TRAINING SET



esempi etichettati

addestro

Modello matematico
(Es. reti neurali)

astraggo il
concetto

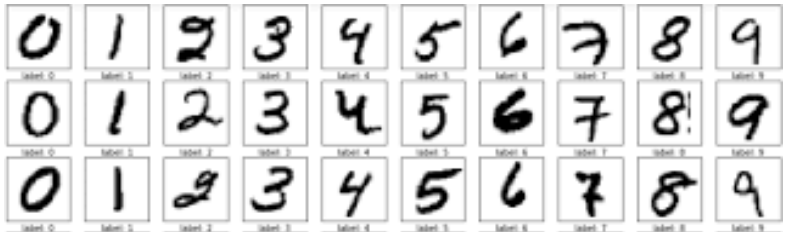
Modello
predittivo

CLASSIFICATORE

MACHINE LEARNING:
sistema di apprendimento supervisionato

scelgo un
MODELLO MATEMATICO

TRAINING SET



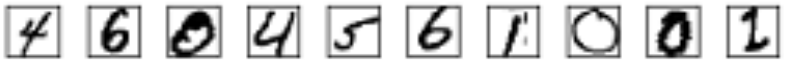
esempi etichettati

addestro

Modello matematico
(Es. reti neurali)

astraggo il
concetto

TEST SET



predire

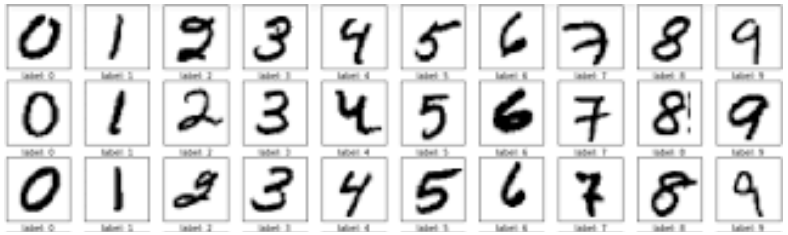
Modello
predittivo

CLASSIFICATORE

MACHINE LEARNING:
sistema di apprendimento supervisionato

scelgo un
MODELLO MATEMATICO

TRAINING SET



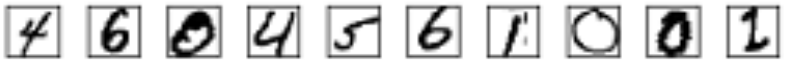
esempi etichettati

addestro

Modello matematico
(Es. reti neurali)

astraggo il
concetto

TEST SET



predire

Modello
predittivo

CLASSIFICATORE

assegno

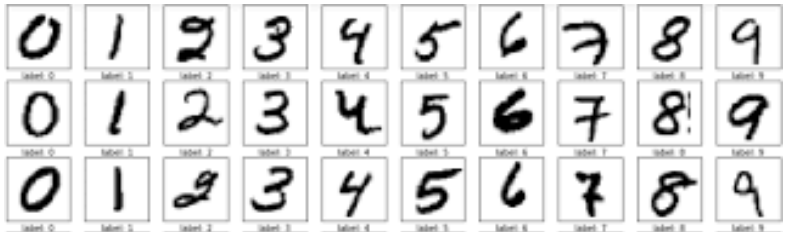
label

- **zero**
- **uno**
- **due**
- **tre**
- **quattro**
- **cinque**
- **sei**
- **sette**
- **otto**
- **nove**

MACHINE LEARNING:
sistema di apprendimento supervisionato

scelgo un
MODELLO MATEMATICO

TRAINING SET



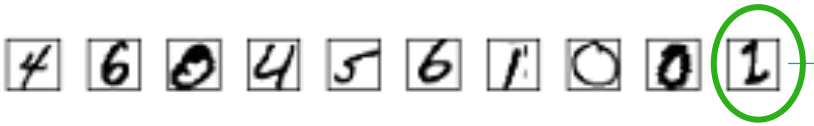
esempi etichettati

addestro

Modello matematico
(Es. reti neurali)

astraggo il
concetto

TEST SET



predire

Modello
predittivo

CLASSIFICATORE

assegno

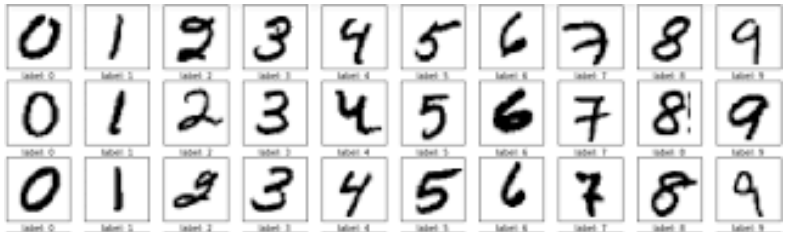
label

- **zero**
- **uno**
- **due**
- **tre**
- **quattro**
- **cinque**
- **sei**
- **sette**
- **otto**
- **nove**

MACHINE LEARNING:
sistema di apprendimento supervisionato

scelgo un
MODELLO MATEMATICO

TRAINING SET



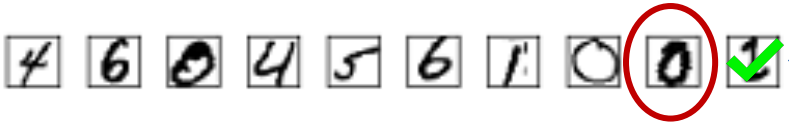
esempi etichettati

addestro

Modello matematico
(Es. reti neurali)

astraggo il
concetto

TEST SET



predire

Modello
predittivo

CLASSIFICATORE

assegno

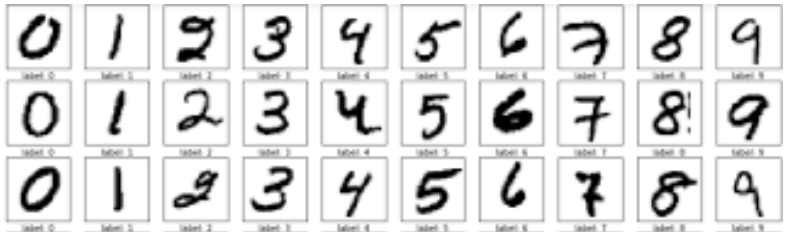
label

- **zero**
- **uno**
- **due**
- **tre**
- **quattro**
- **cinque**
- **sei**
- **sette**
- **otto**
- **nove**

MACHINE LEARNING:
sistema di apprendimento supervisionato

scelgo un
MODELLO MATEMATICO

TRAINING SET



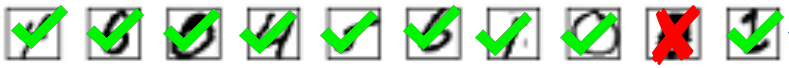
esempi etichettati

addestro

Modello matematico
(Es. reti neurali)

astraggo il
concetto

TEST SET



90%

predire

Modello
predittivo

CLASSIFICATORE

assegno

label

- **zero**
- **uno**
- **due**
- **tre**
- **quattro**
- **cinque**
- **sei**
- **sette**
- **otto**
- **nove**

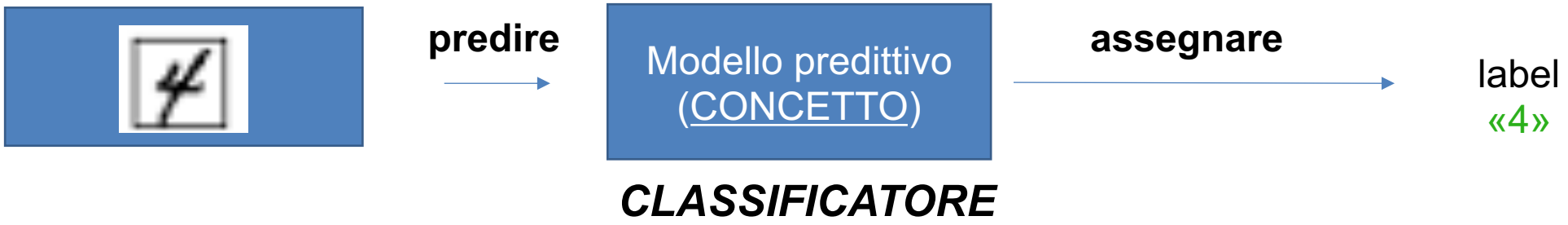
LE FASI DEL MACHINE LEARNING

1. Training
2. Prediction

LA FASE DI TRAINING

Usando gli **esempi** (training set), addestro il modello matematico per **astrarre** il **concetto** (modello predittivo)

LA FASE DI PREDICTION



il modello predittivo o ancora meglio il

classificatore predice valori futuri

assegnando una label

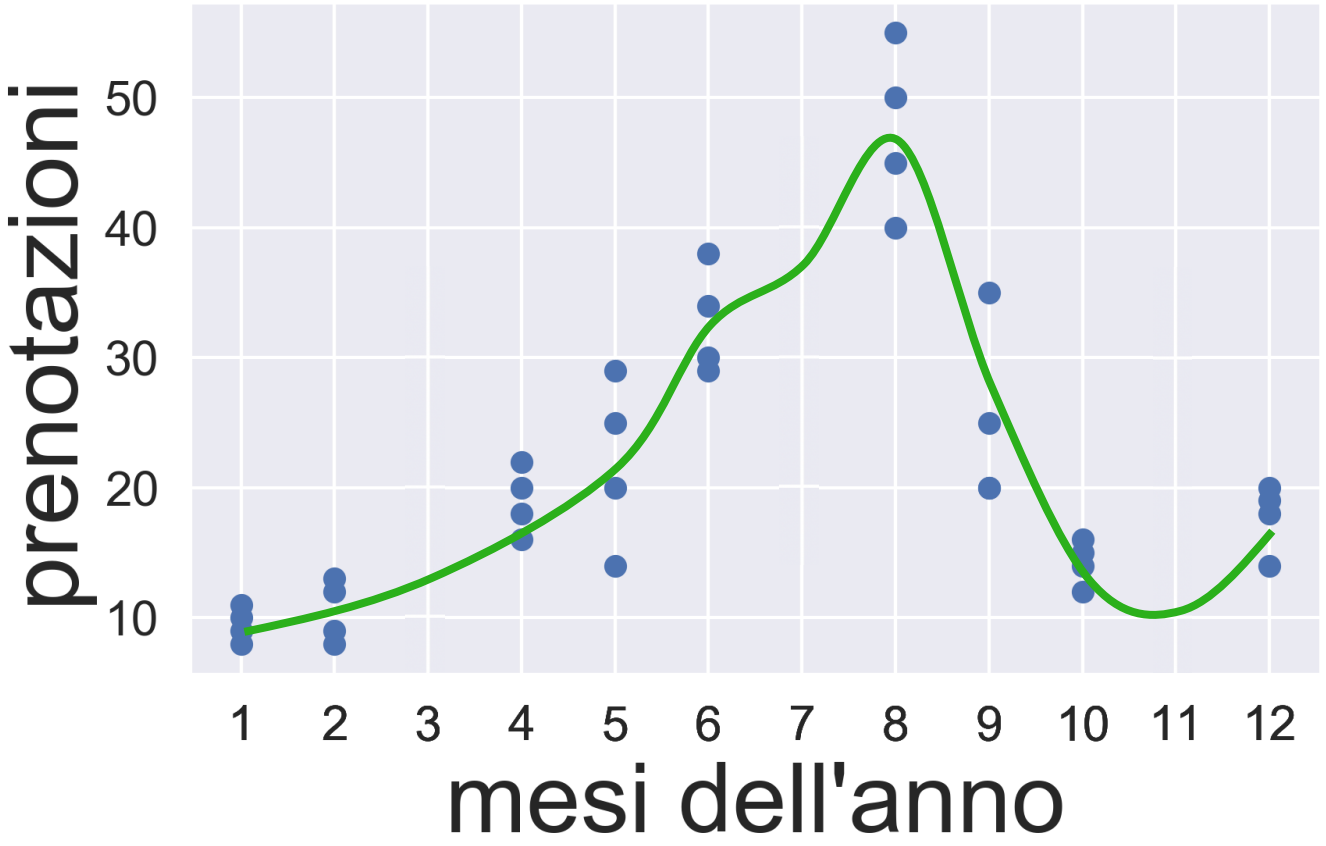
LE FASI DEL MACHINE LEARNING

Apprendimento supervisionato

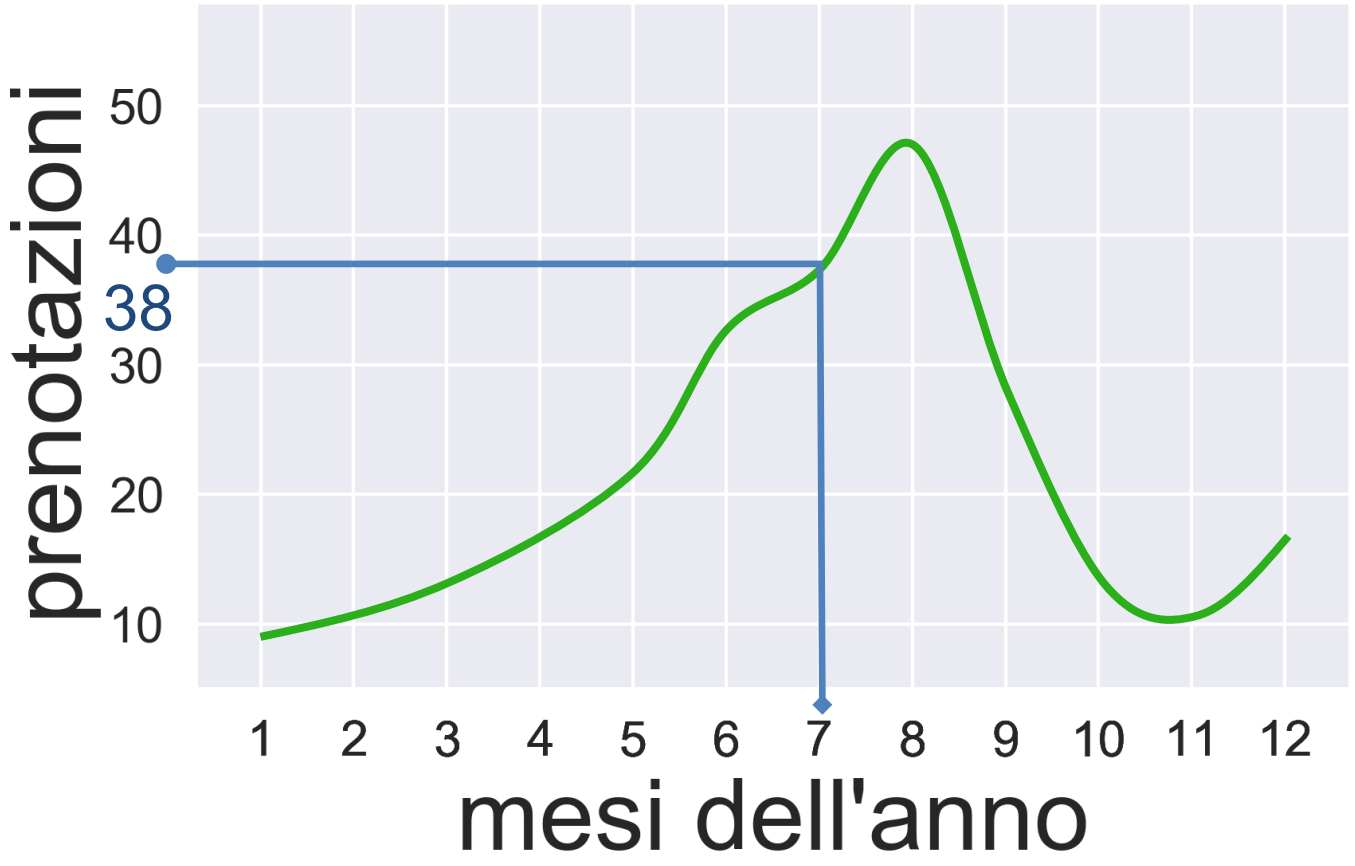
1. Training
2. Prediction

LA MAGIA DIETRO IL MACHINE LEARNING

MACHINE LEARNING



MACHINE LEARNING



UN LUNGO VIAGGIO VERSO LA «COMPUTER VISION»

ESEMPIO

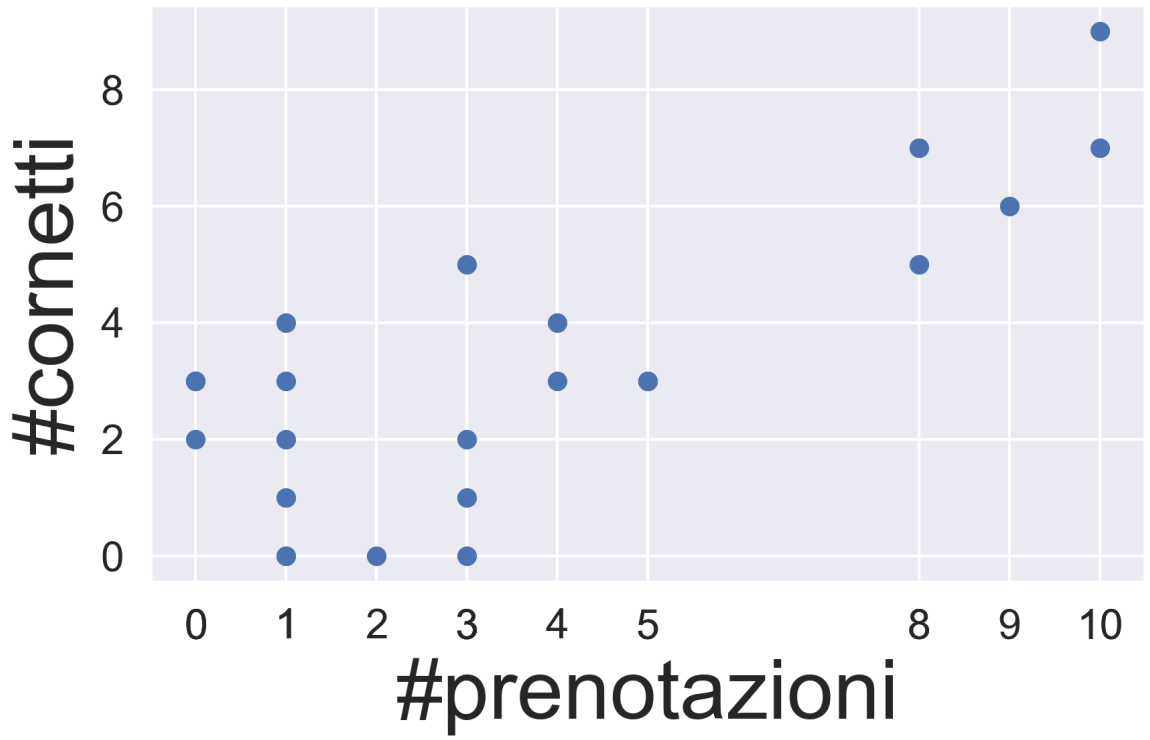
QUANTI CORNETTI?

ESEMPIO

TRAINING SET

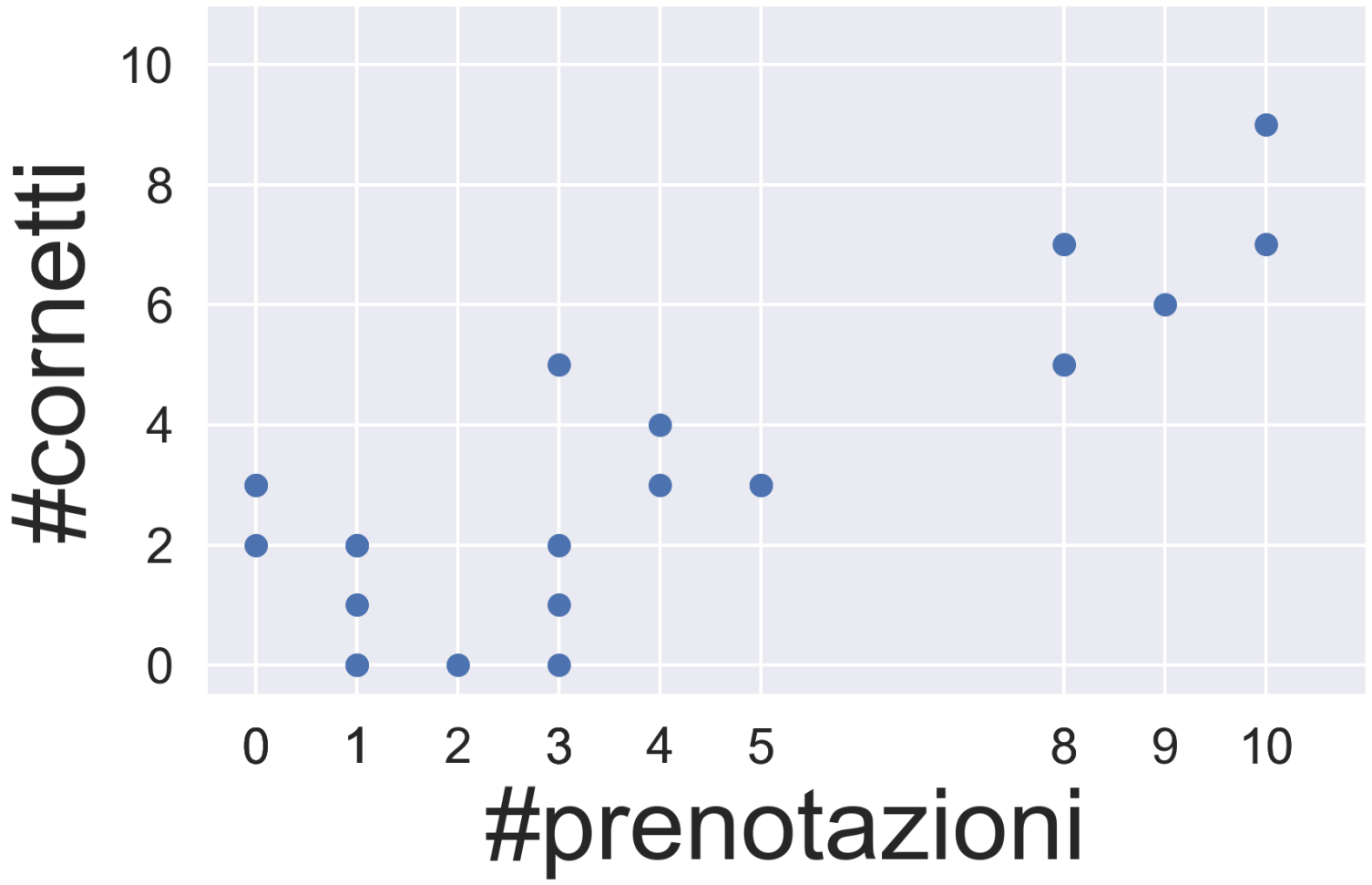
X = #PRENOTAZIONI	Y = #CORNETTI
1	4
1	3
3	2
3	1
5	3
8	7
10	9
4	3
1	1
1	0
1	0
1	2
…	…

```
import numpy as np
X, Y = np.loadtxt(«cornetti.txt», skiprows=1, unpack=True)
```

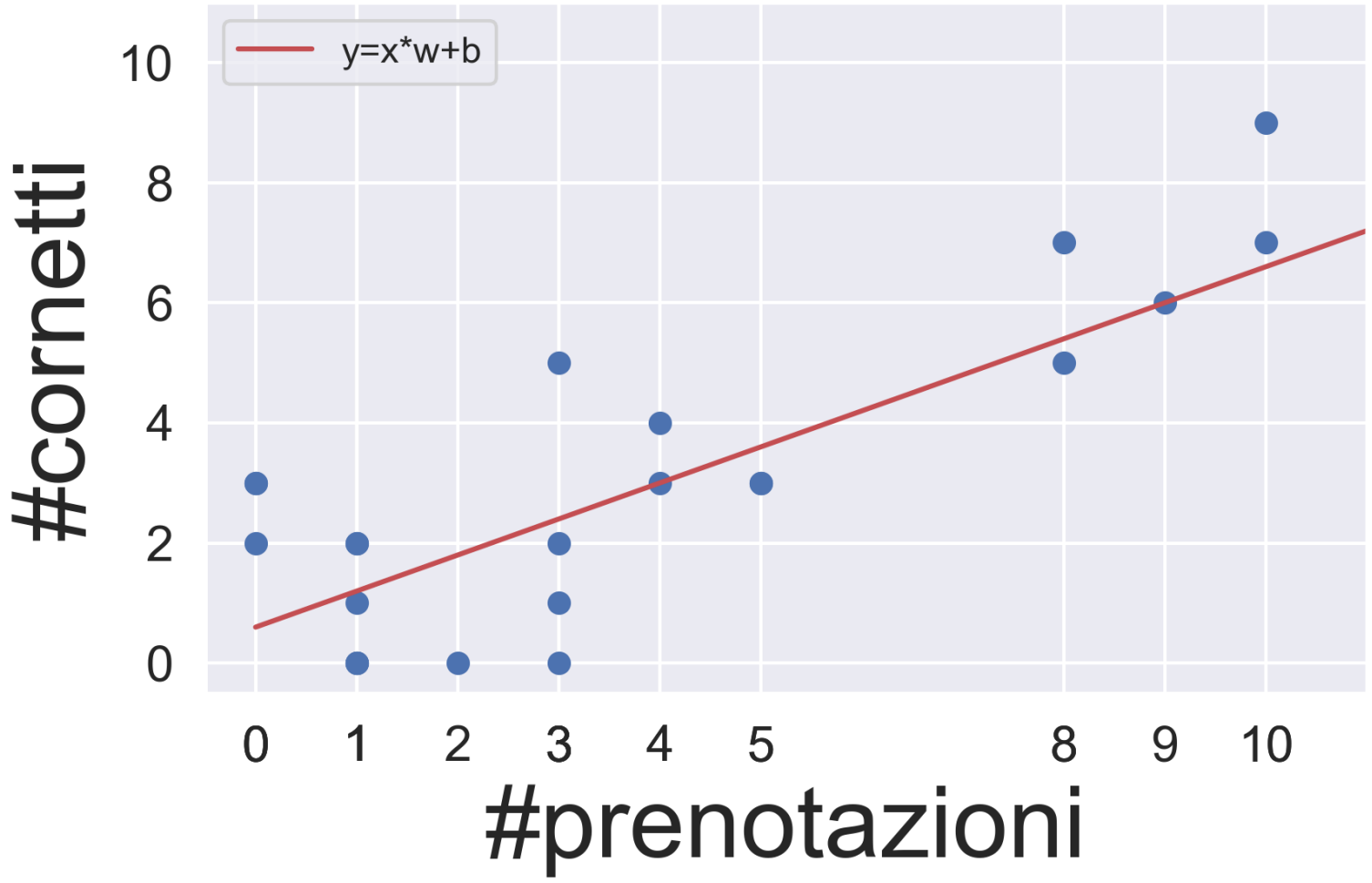



```
import matplotlib.pyplot as plt
plt.plot(X, Y, «bo»)
```

OBIETTIVO?



ESEMPIO



REGRESSIONE LINEARE

y=x*<u>**w**</u> + <u>**b**</u>

LE FASI DEL MACHINE LEARNING

Apprendimento supervisionato

1. Training
2. Prediction

1. Fase di Training

X = #PRENOTAZIONI	Y = #CORNETTI
1	4
1	3
3	2
..	..

Training set (ESEMPI)

addestro

Modello matematico (ASTRAE)

REGRESSIONE LINEARE:
$y=x*\mathbf{w} + \mathbf{b}$

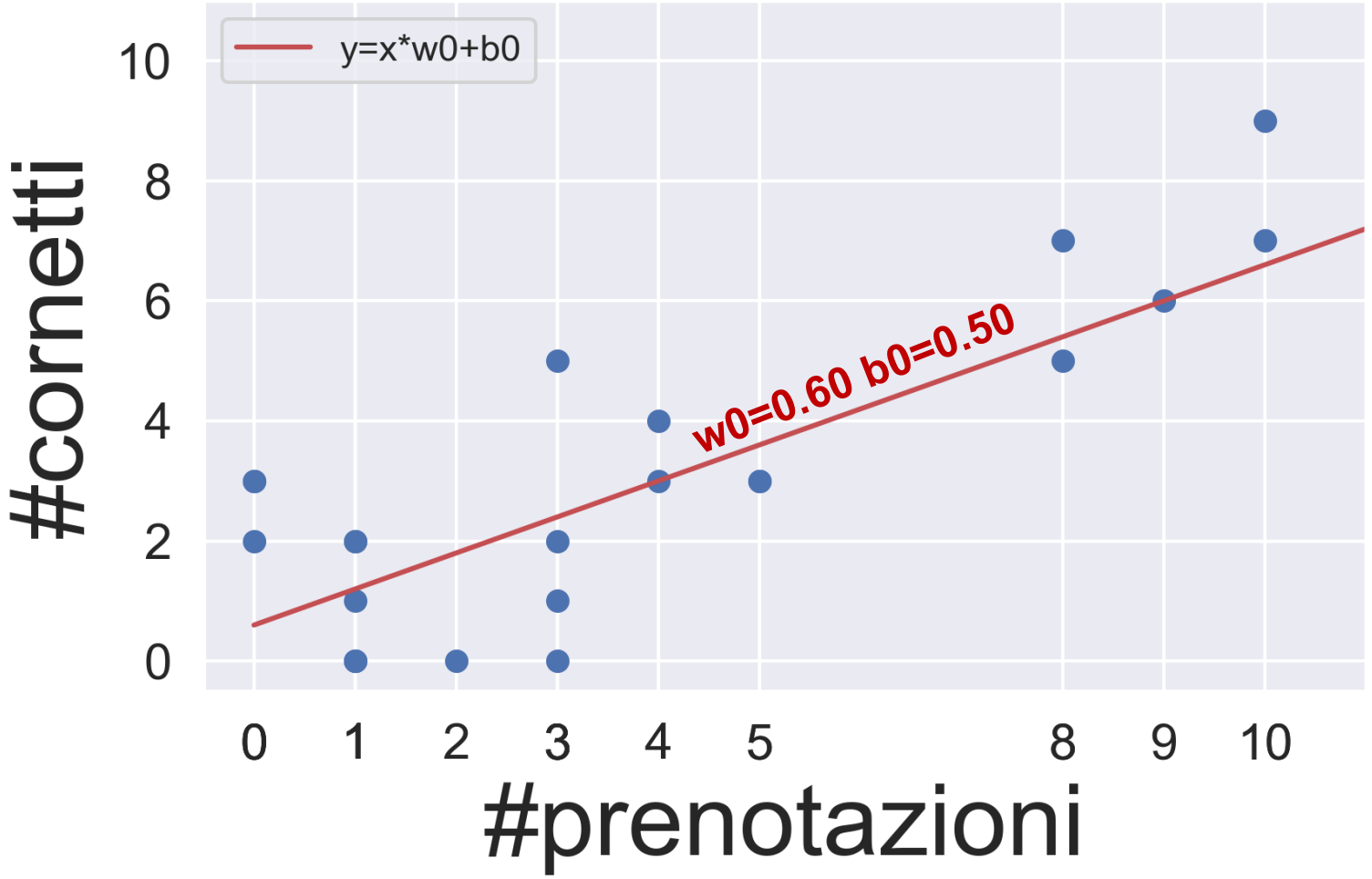
astrarre

Modello predittivo (CONCETTO)

$y=x*\mathbf{w0}+ \mathbf{b0}$

w0=0.60 **b0**=0.50

1. Fase di Training



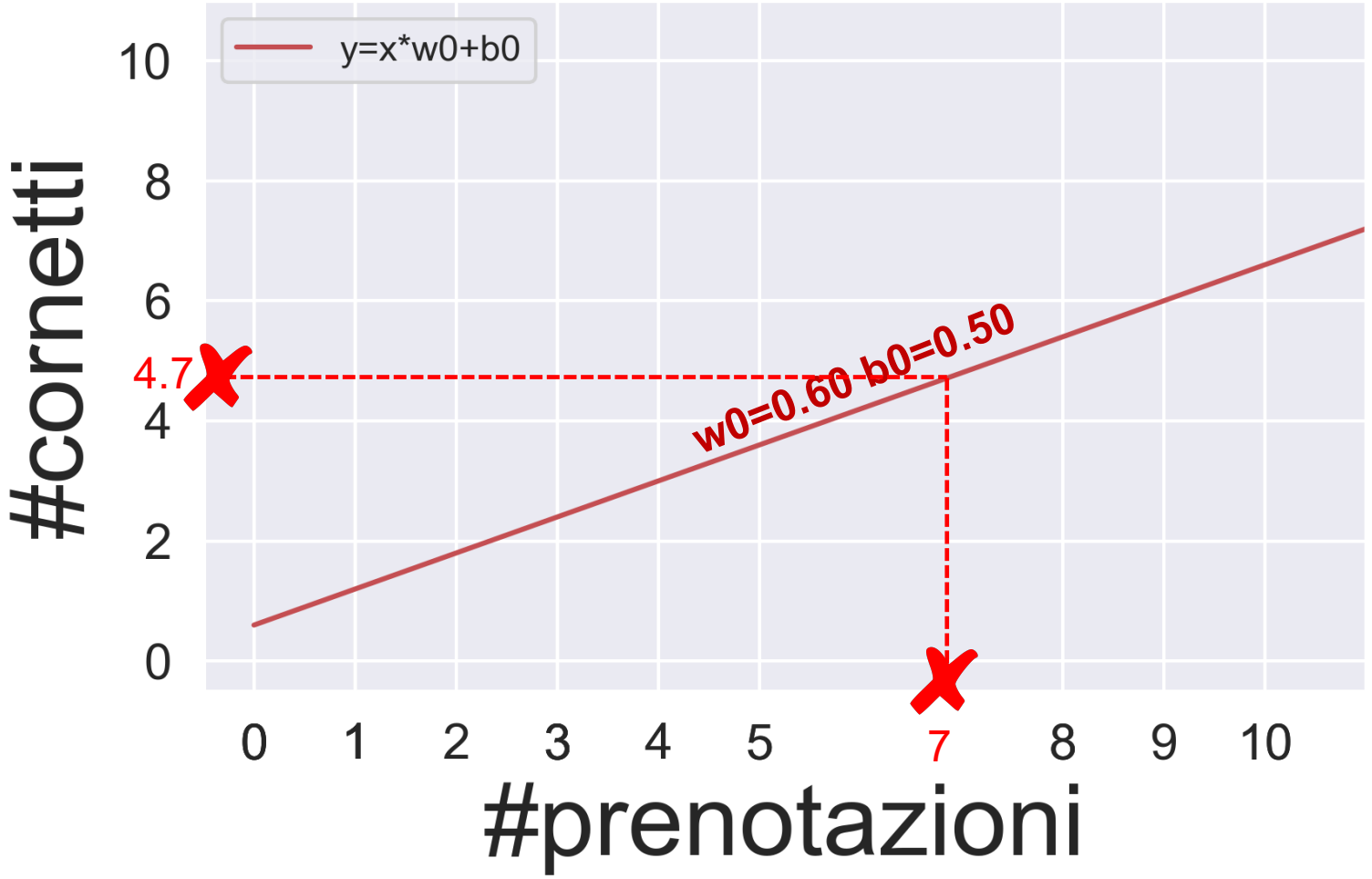
Modello predittivo (CONCETTO)

y=x*<u>**w0**</u>+ <u>**b0**</u>

w0=0.60 **b0**=0.50

Traccio una linea che meglio approssimano i miei esempi

2. Fase di Prediction



x = 7 → Modello predittivo (CONCETTO) → y = 4.7

y=x***w0**+ **b0**

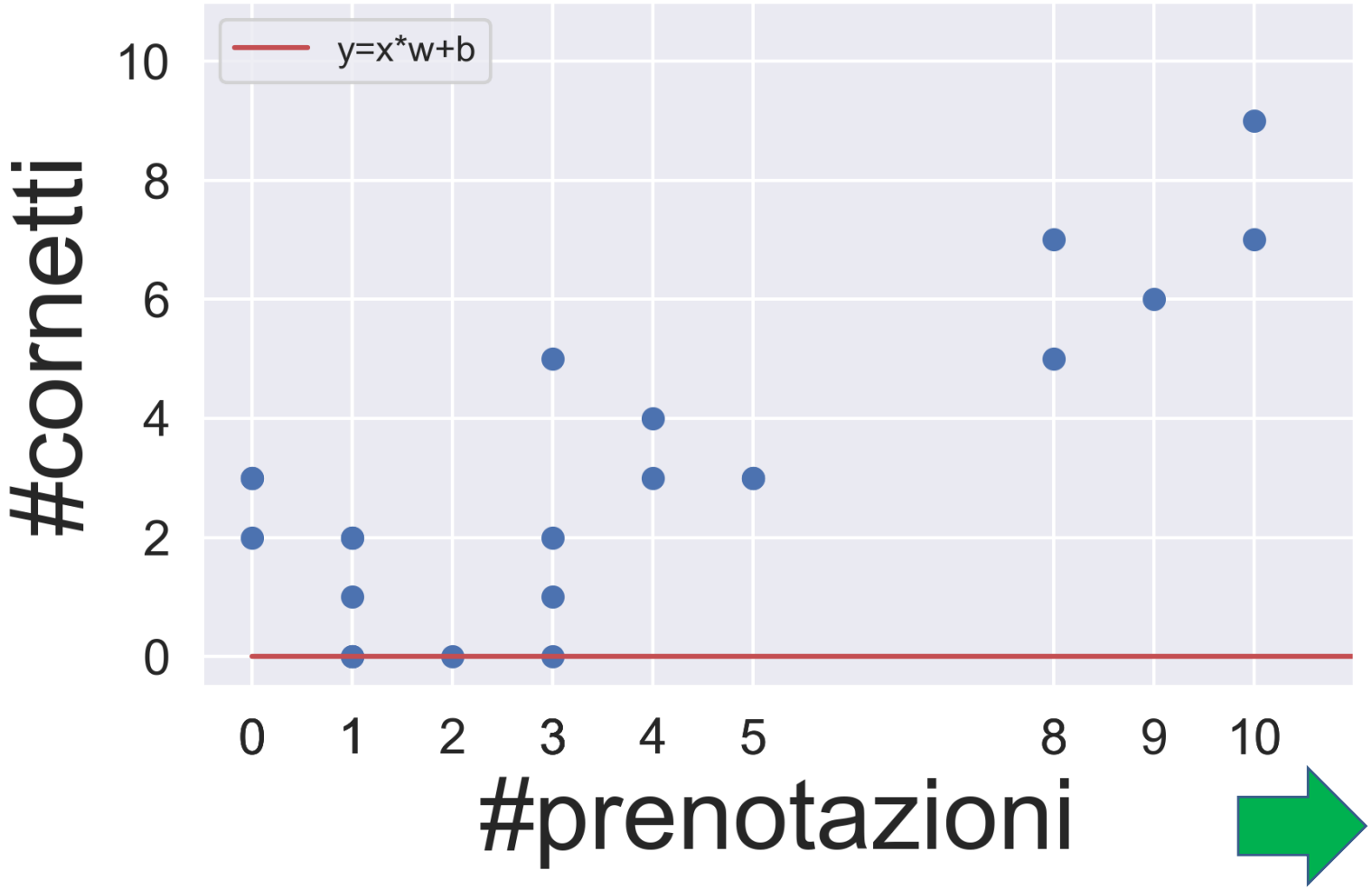
IMPLEMENTIAMO LA **FASE DI PREDICTION**

$y = x * \mathbf{w} + \mathbf{b}$

```
def prediction(X,w,b):
    return X*w+b
```

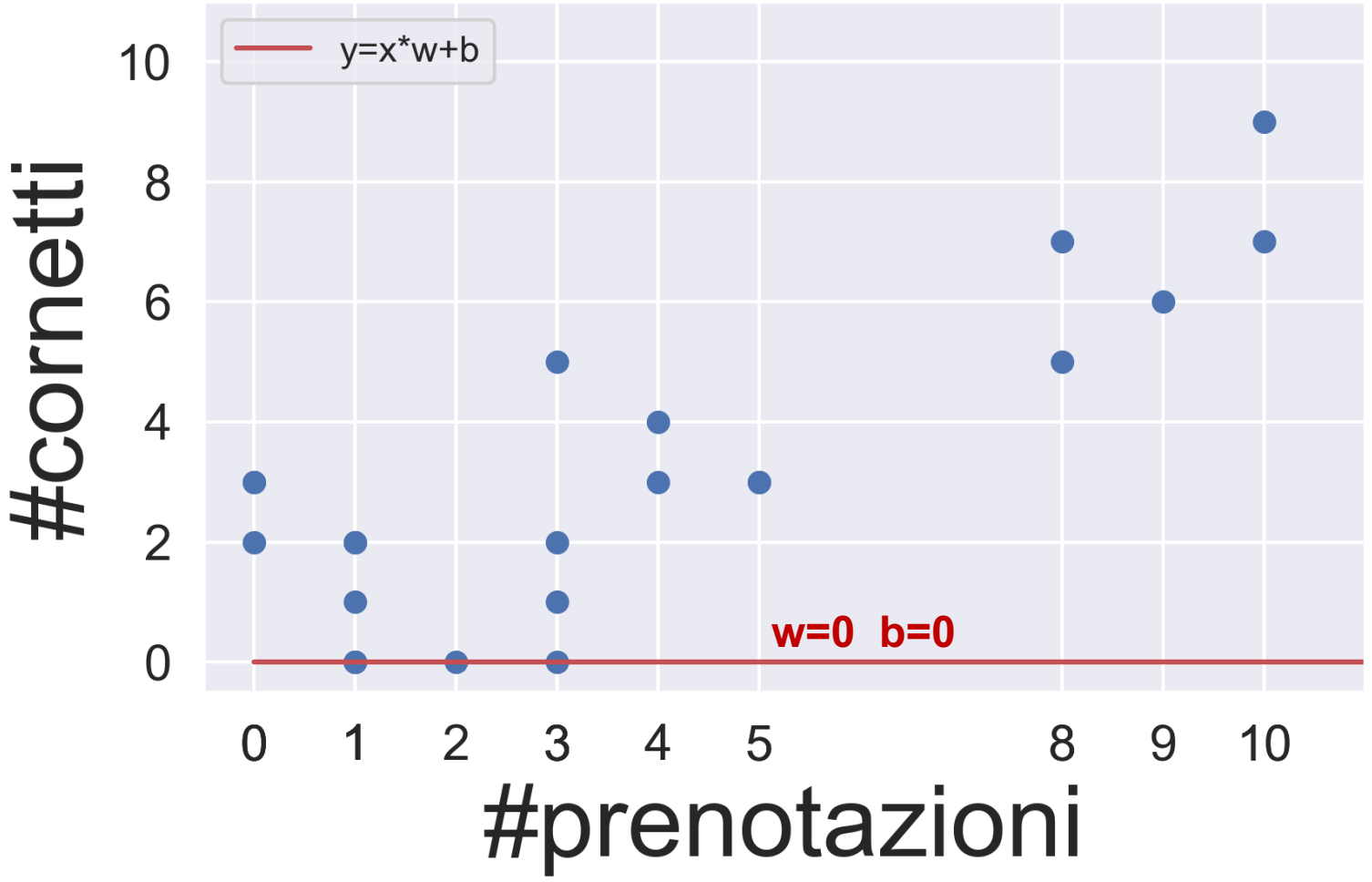
IMPLEMENTIAMO LA **FASE TRAINING**

STRATEGIA PER TROVARE LA «BEST LINE» ?



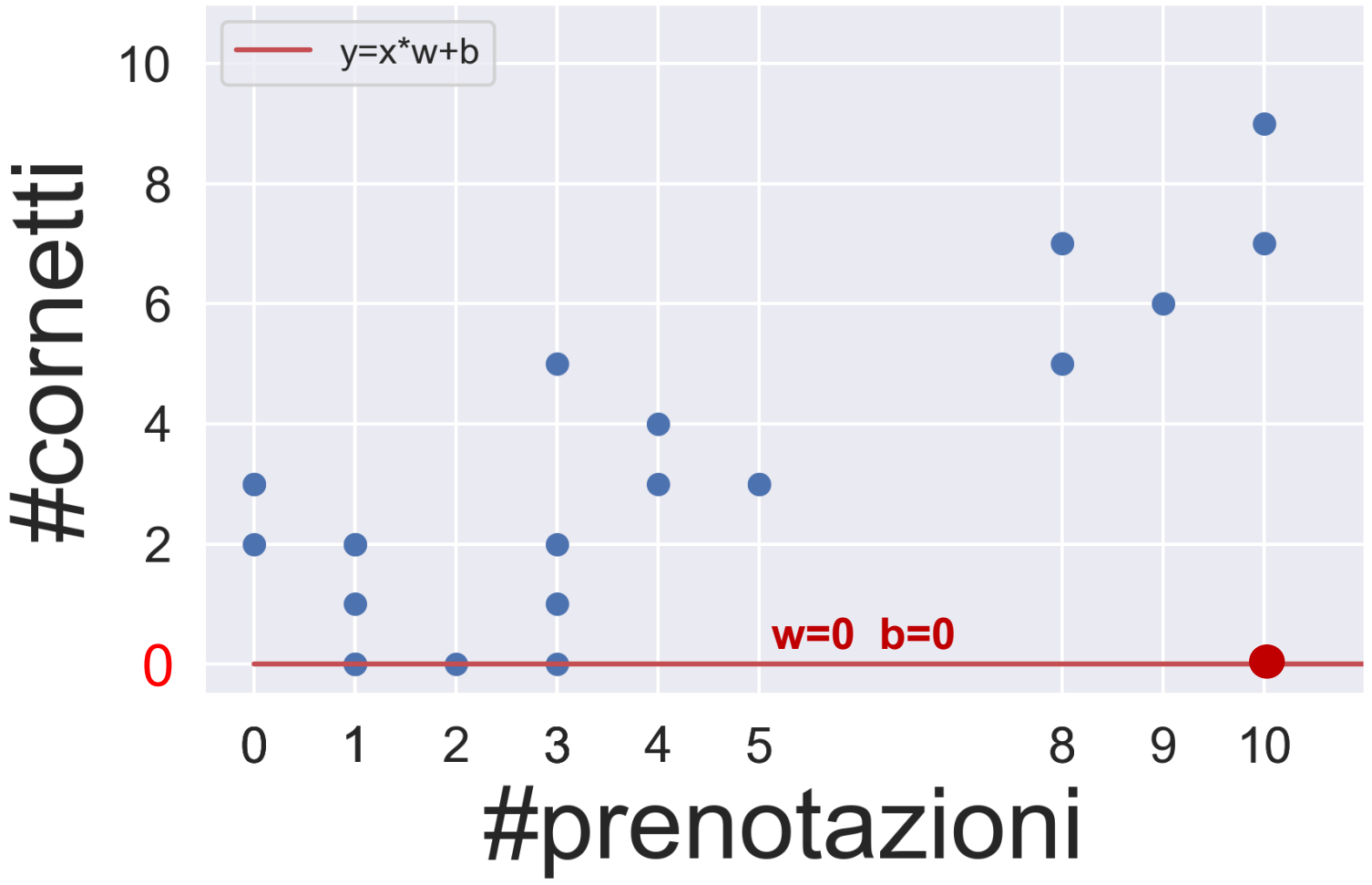
OBIETTIVO:
trovare i valori di **w** e **b** della retta che **minimizzano** la **loss**

STRATEGIA PER TROVARE LA «BEST LINE» ?



OBIETTIVO:
trovare i valori di **w** e **b** della retta che **minimizzano** la **loss**

STRATEGIA PER TROVARE LA «BEST LINE» ?

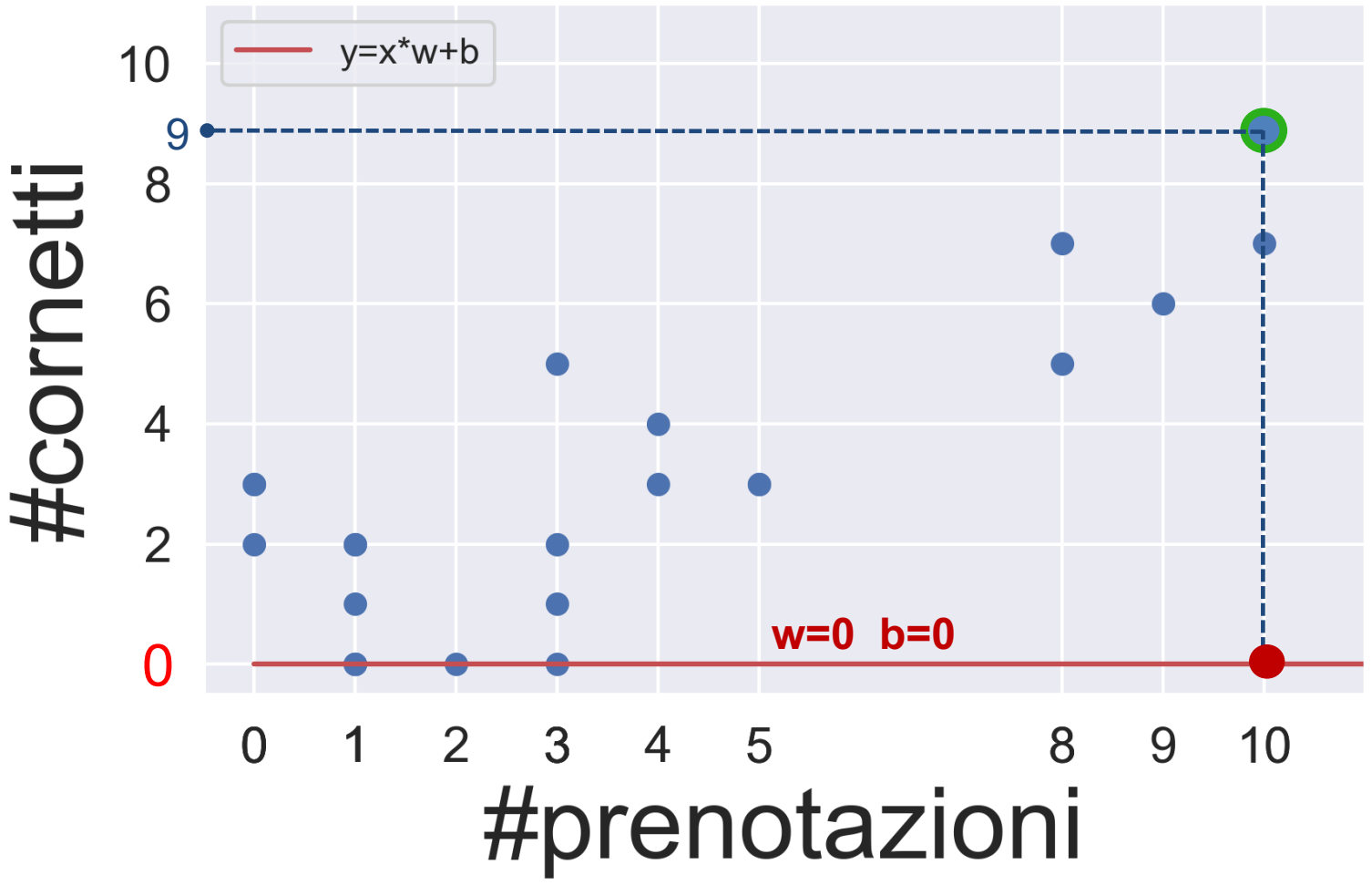


Prima iterazione:
w = 0
b = 0

```
def loss(X,Y,w,b):
```

OBIETTIVO:
trovare i valori di **w** e **b** della retta che **minimizzano** la **loss**

STRATEGIA PER TROVARE LA «BEST LINE» ?

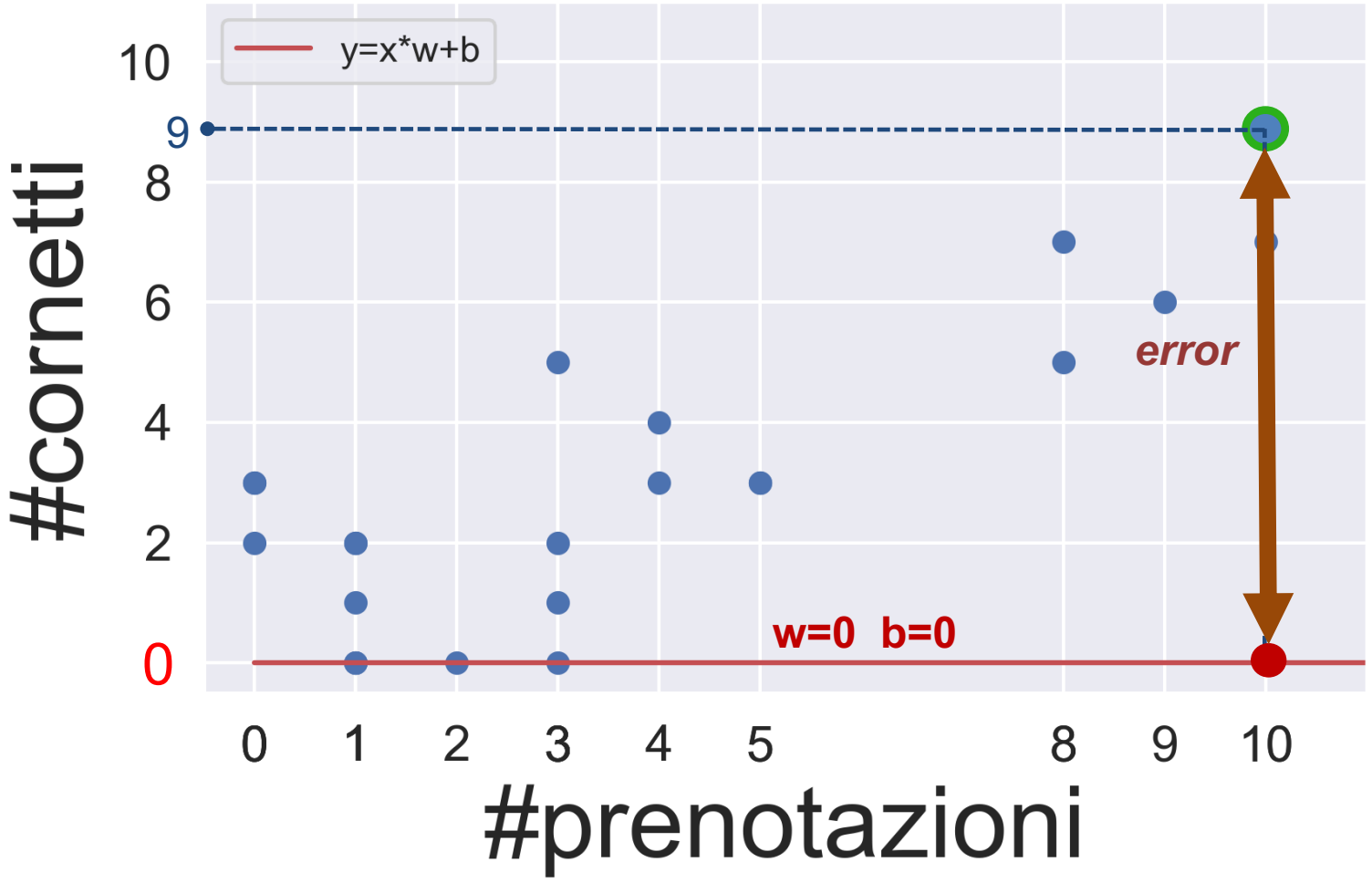


Prima iterazione:
w = 0
b = 0

```
def loss(X,Y,w,b):
```

OBIETTIVO:
trovare i valori di **w** e **b** della retta che **minimizzano** la **loss**

STRATEGIA PER TROVARE LA «BEST LINE» ?

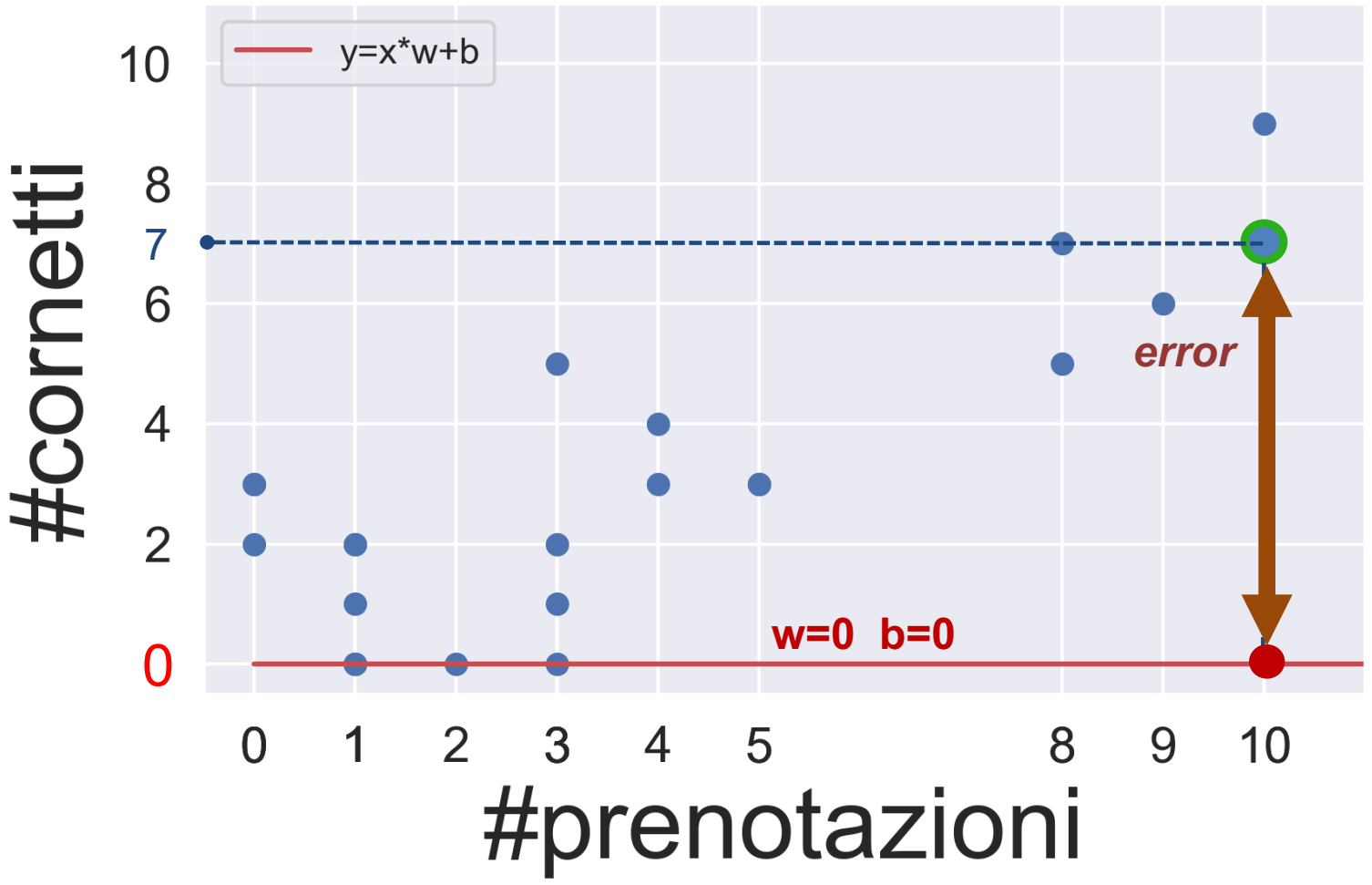


Prima iterazione:
w = 0
b = 0

```
def loss(X,Y,w,b):
    error=prediction(X,w,b) - Y
```

OBIETTIVO:
trovare i valori di **w** e **b** che **minimizzano** la **loss**

STRATEGIA PER TROVARE LA «BEST LINE» ?

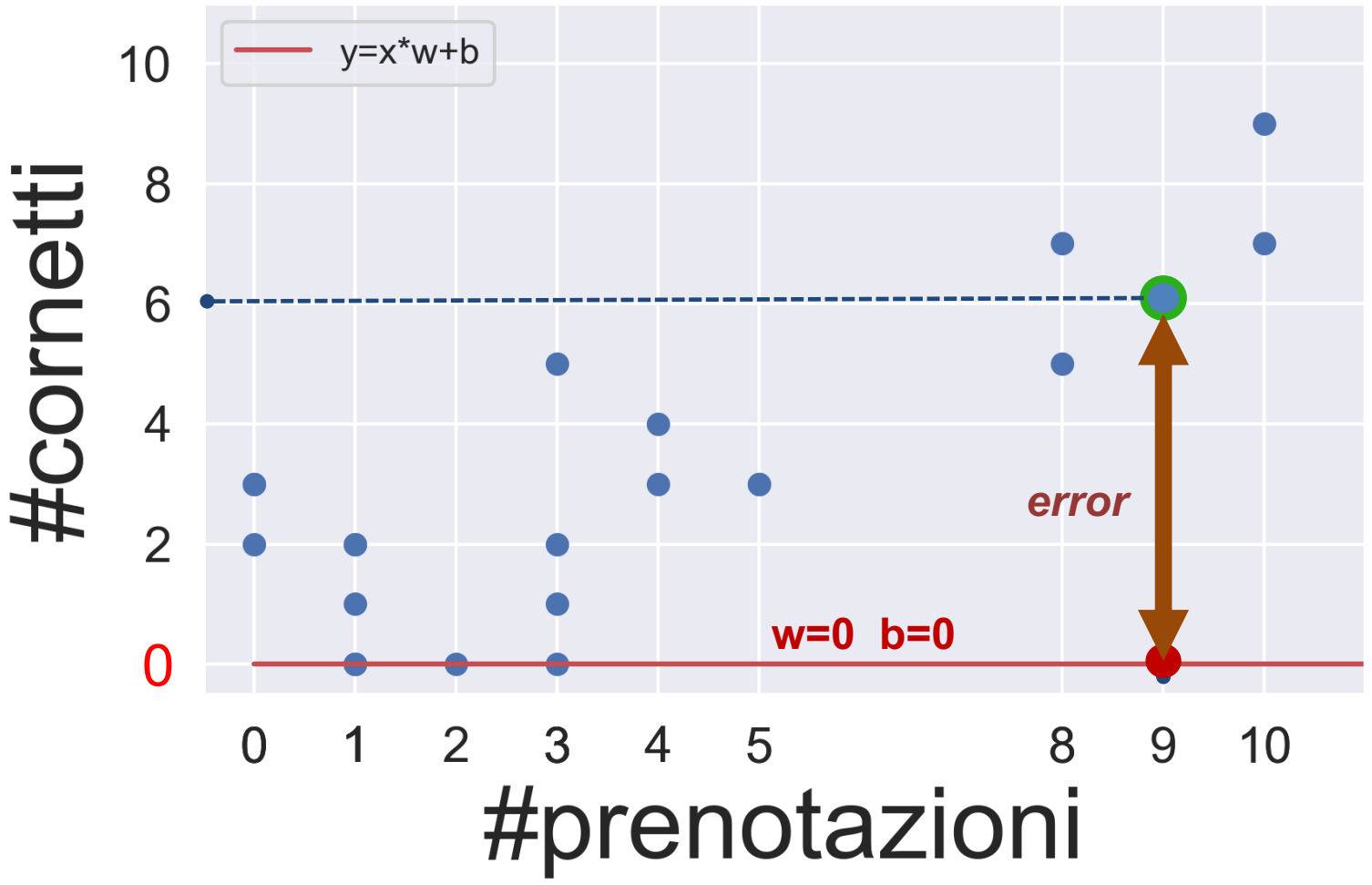


Prima iterazione:
w = 0
b = 0

```
def loss(X,Y,w,b):
    error=prediction(X,w,b) - Y
```

OBIETTIVO:
trovare i valori di **w** e **b** della retta che **minimizzano** la **loss**

STRATEGIA PER TROVARE LA «BEST LINE» ?

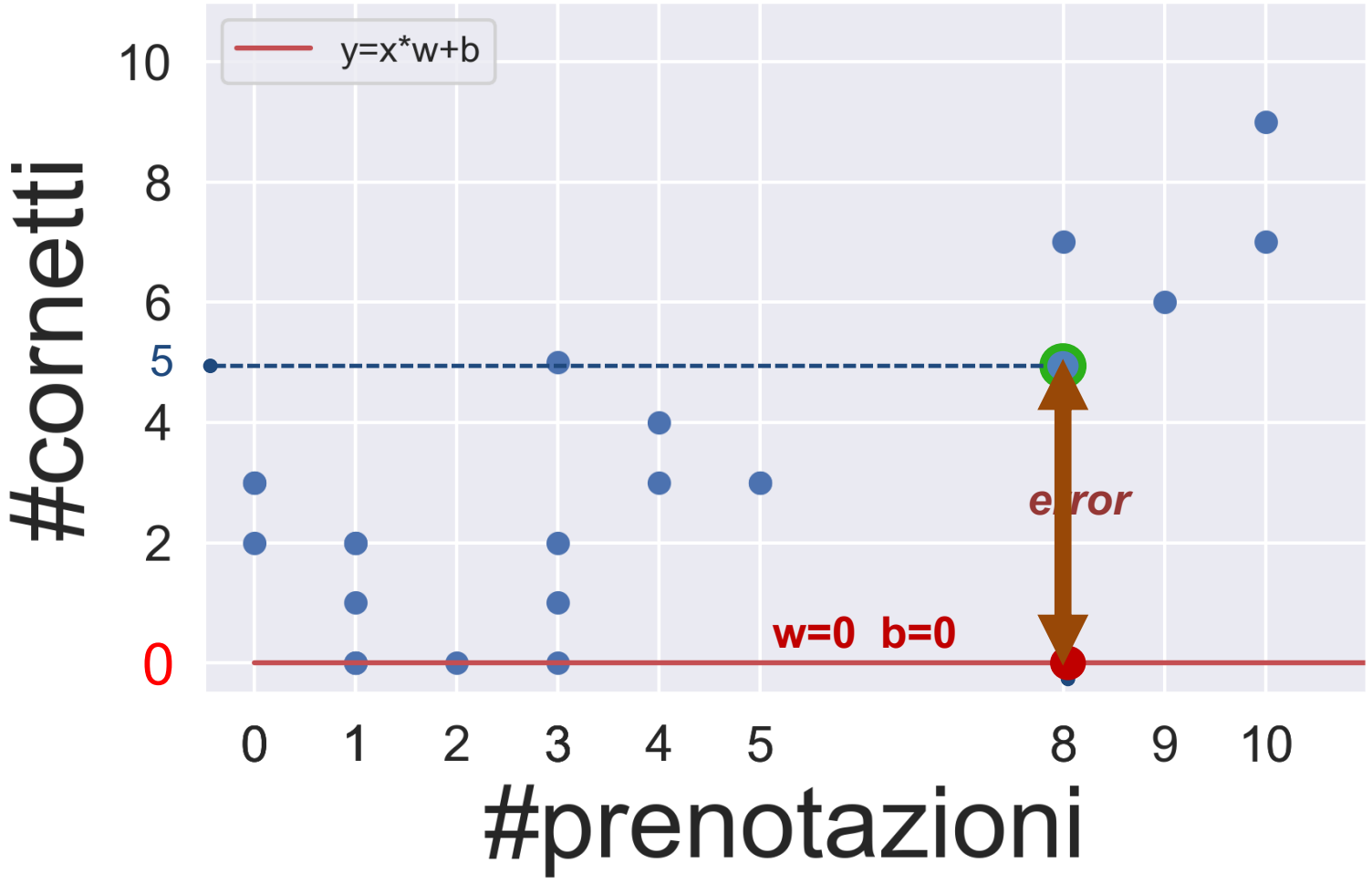


Prima iterazione:
w = 0
b = 0

```
def loss(X,Y,w,b):
    error=prediction(X,w,b) - Y
```

OBIETTIVO:
trovare i valori di **w** e **b** della retta che **minimizzano** la **loss**

STRATEGIA PER TROVARE LA «BEST LINE» ?

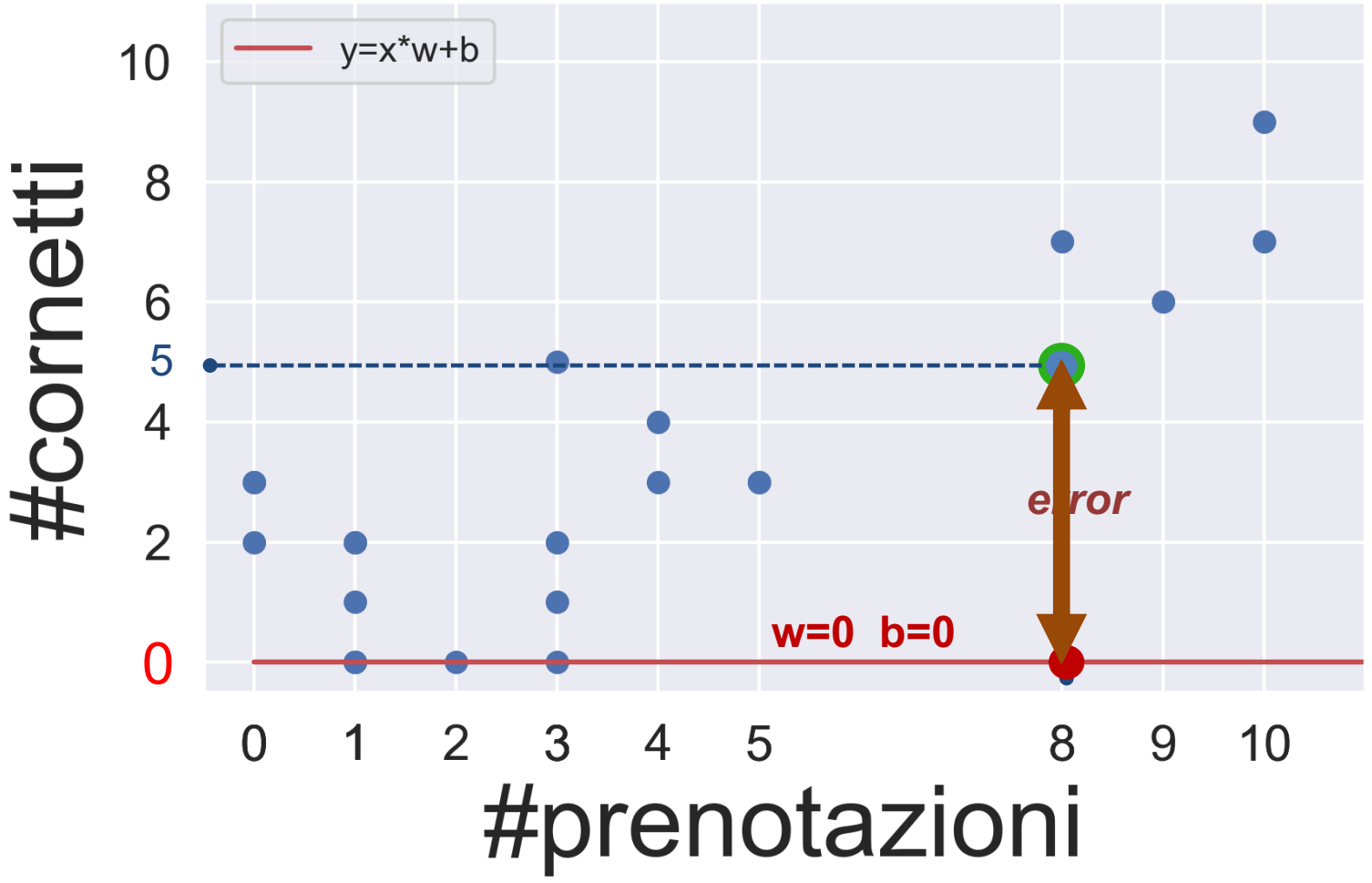


Prima iterazione:
w = 0
b = 0

```
def loss(X,Y,w,b):
    error=prediction(X,w,b) - Y
```

OBIETTIVO:
trovare i valori di **w** e **b** della retta che **minimizzano** la **loss**

STRATEGIA PER TROVARE LA «BEST LINE» ?

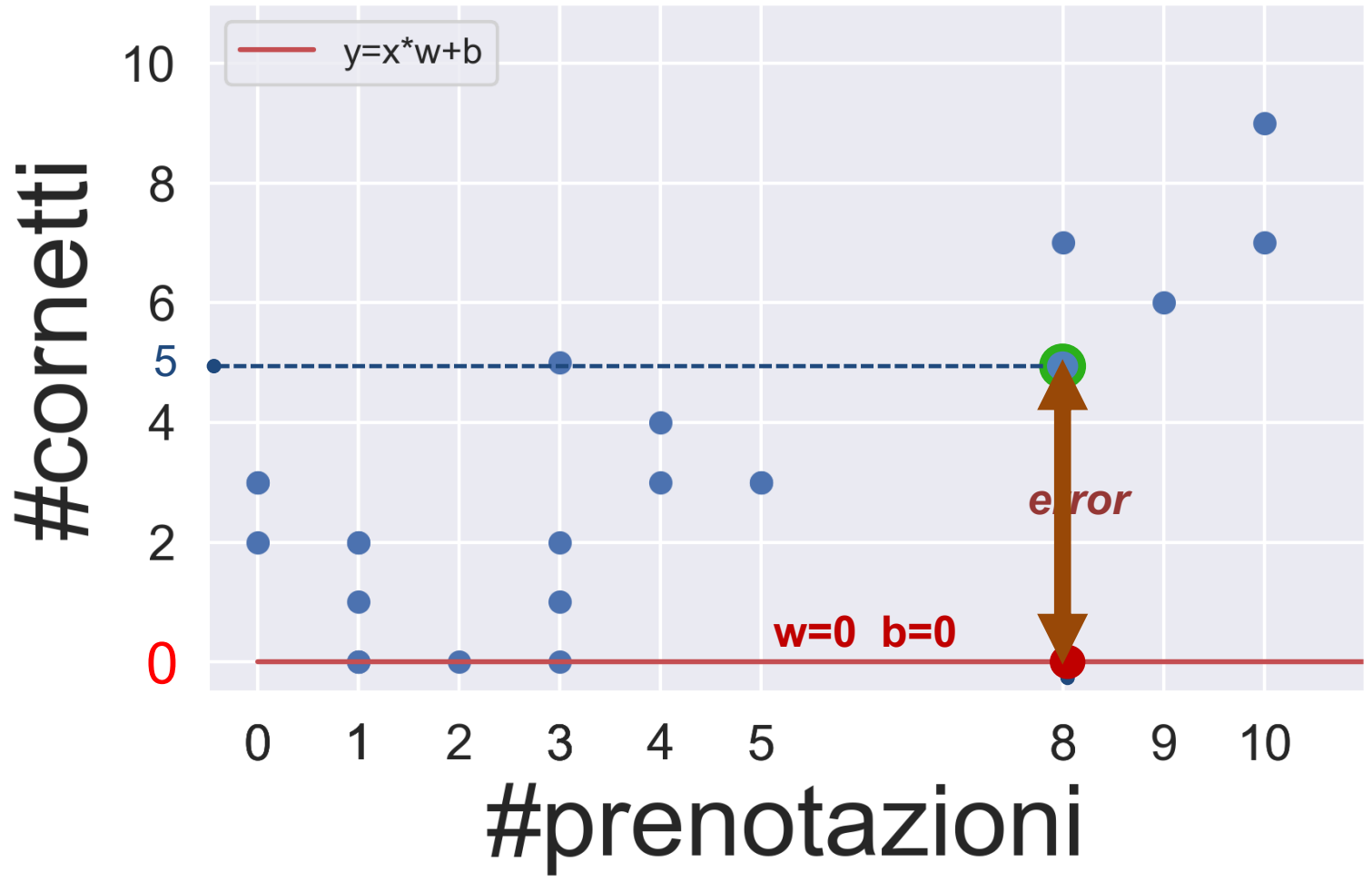


Prima iterazione:
w = 0
b = 0

```
def loss(X,Y,w,b):
    error=prediction(X,w,b) - Y
    squared_error = error**2
```

OBIETTIVO:
trovare i valori di **w** e **b** della retta che **minimizzano** la **loss**

STRATEGIA PER TROVARE LA «BEST LINE» ?



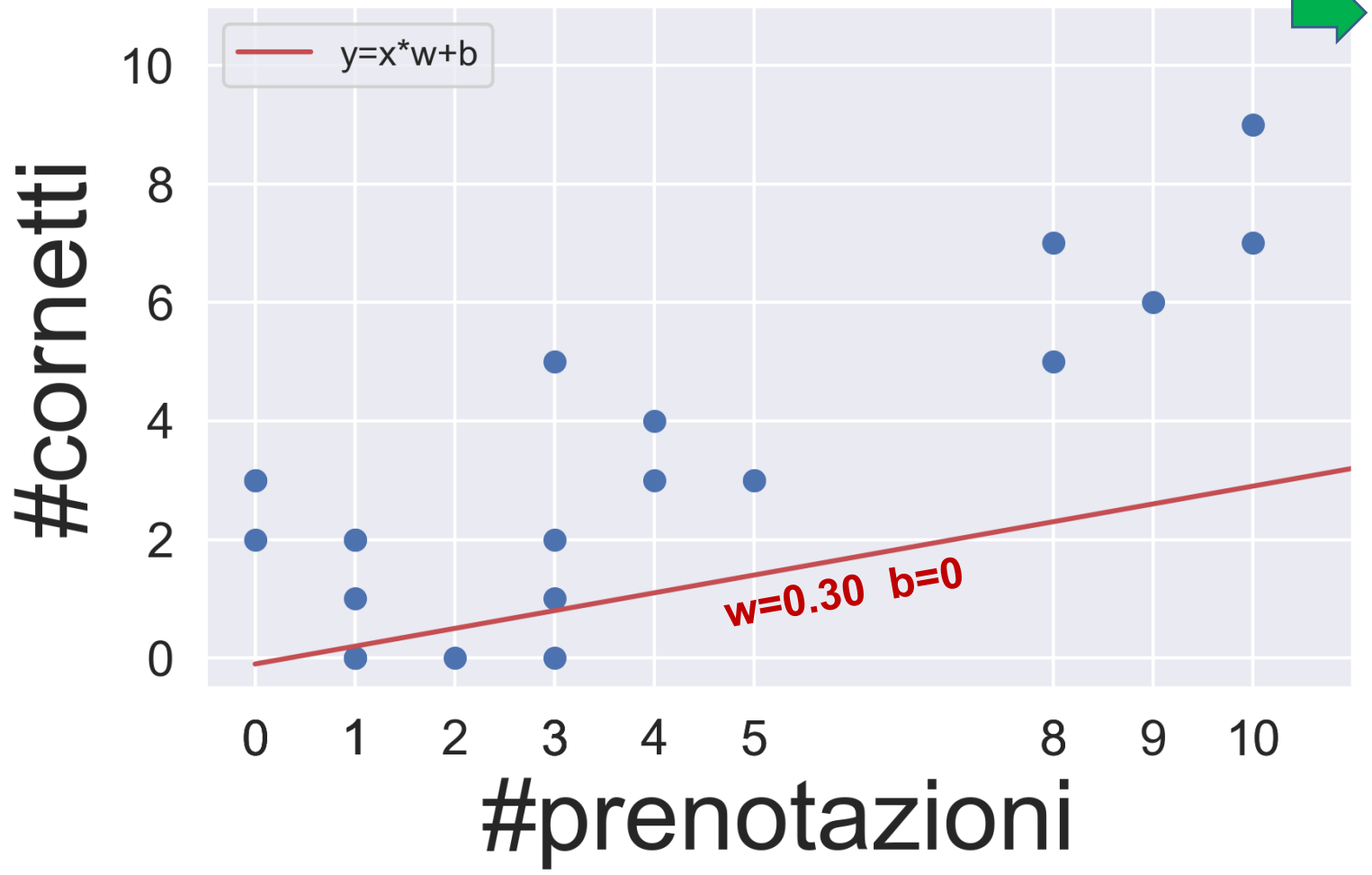
Prima iterazione:
w = 0
b = 0

```
def loss(X,Y,w,b):
    error=prediction(X,w,b) - Y
    squared_error = error**2
    return np.average(squared_error)
```

loss=810

OBIETTIVO:
trovare i valori di **w** e **b** della retta che **minimizzano** la **loss**

STRATEGIA PER TROVARE LA «BEST LINE» ?

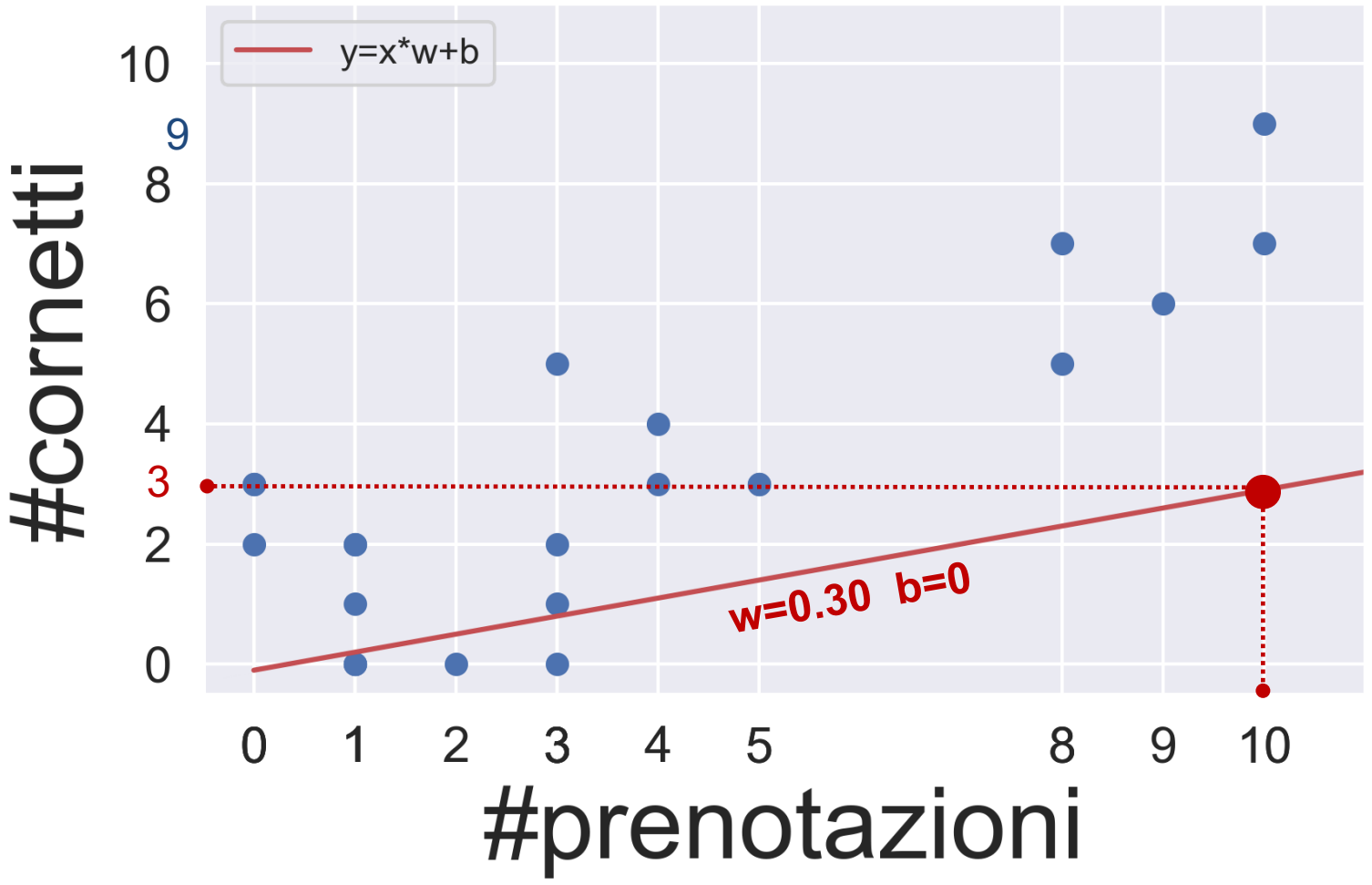


Seconda Iterazione
w= 0.30 (incrementata di 0.30)
b =0

```
def loss(X,Y,w,b):
```

OBIETTIVO:
trovare i valori di **w** e **b** che **minimizzano** la **loss**

STRATEGIA PER TROVARE LA «BEST LINE» ?

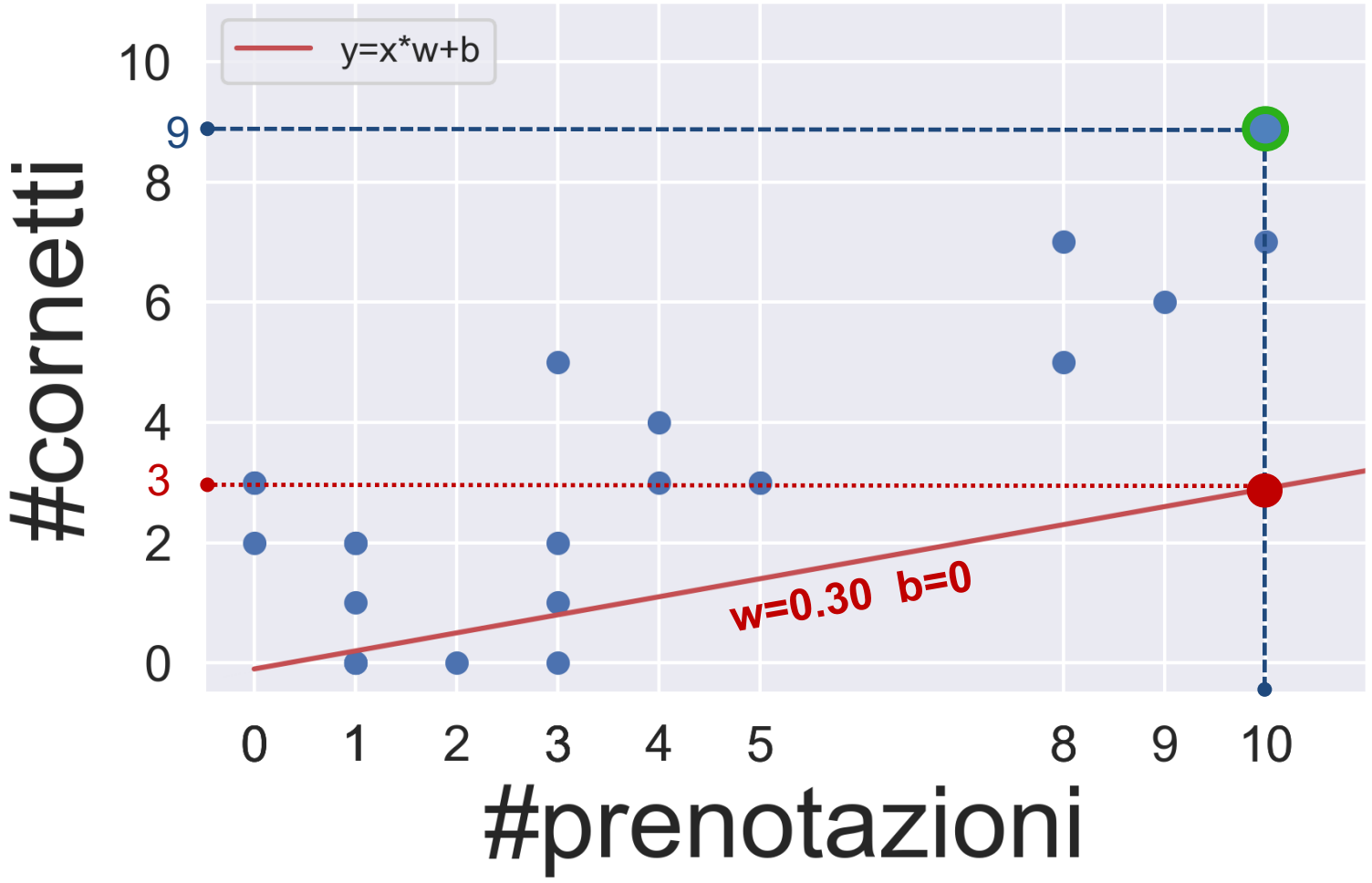


Seconda Iterazione
w= 0.30
b =0

```
def loss(X,Y,w,b):
```

OBIETTIVO:
trovare i valori di **w** e **b** della retta che **minimizzano** la **loss**

STRATEGIA PER TROVARE LA «BEST LINE» ? w e b

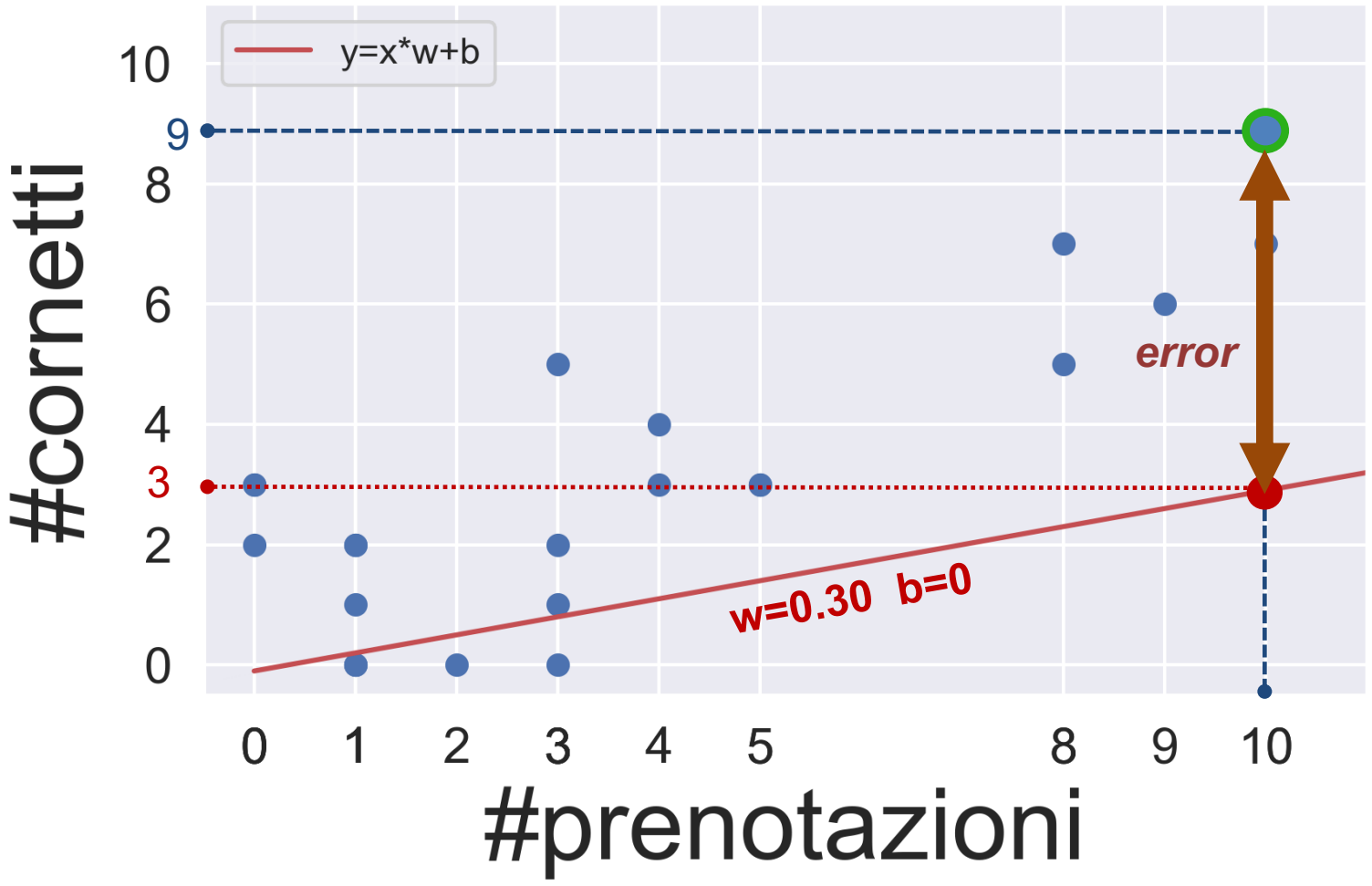


Seconda Iterazione
w= 0.30
b =0

```
def loss(X,Y,w,b):
```

OBIETTIVO:
trovare i valori di **w** e **b** della retta che **minimizzano** la **loss**

STRATEGIA PER TROVARE LA «BEST LINE» ?

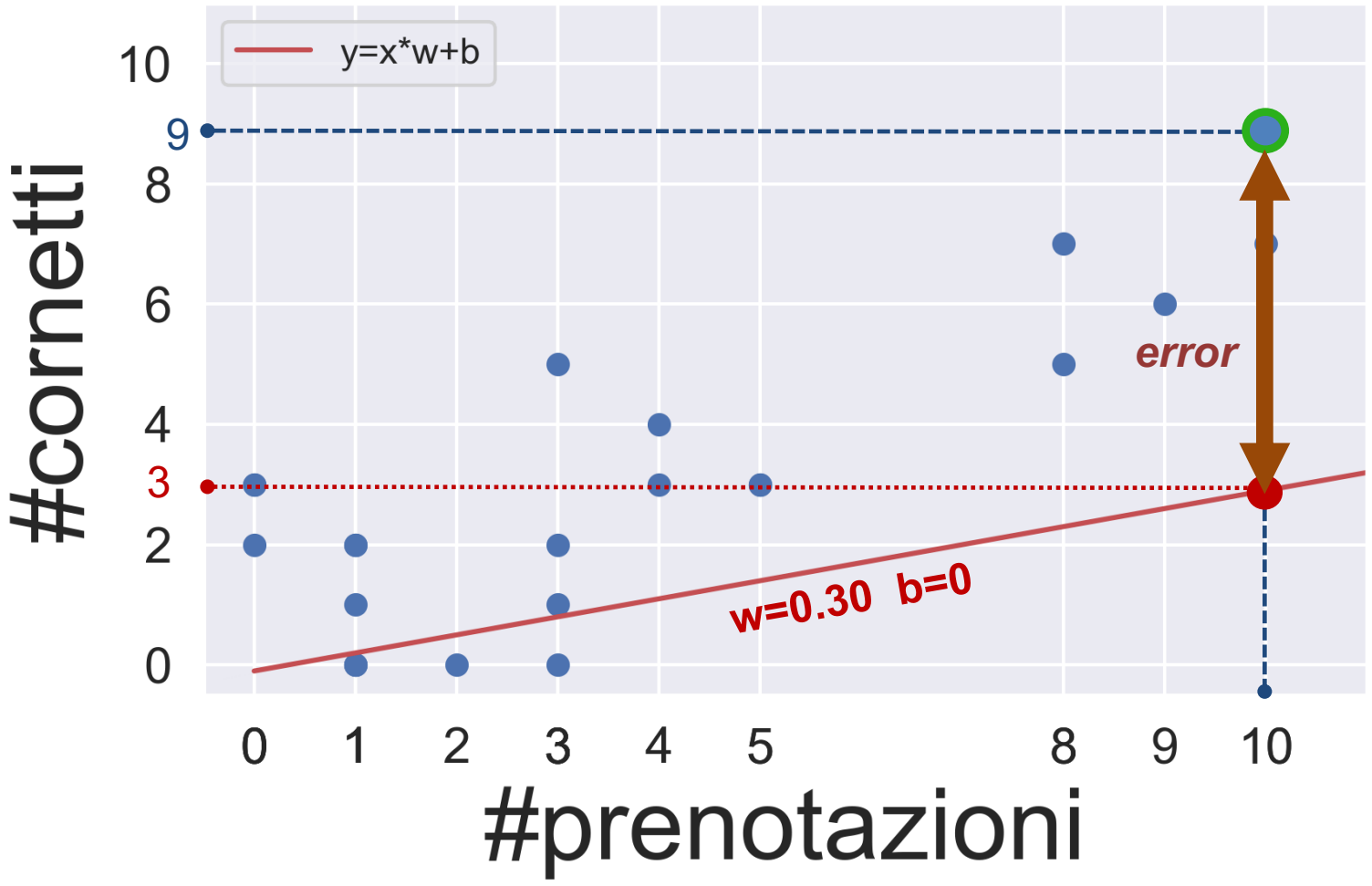


Seconda Iterazione
w= 0.30
b =0

```
def loss(X,Y,w,b):
    error=prediction(X,w,b) - Y
```

OBIETTIVO:
trovare i valori di **w** e **b** della retta che **minimizzano** la **loss**

STRATEGIA PER TROVARE LA «BEST LINE» ?

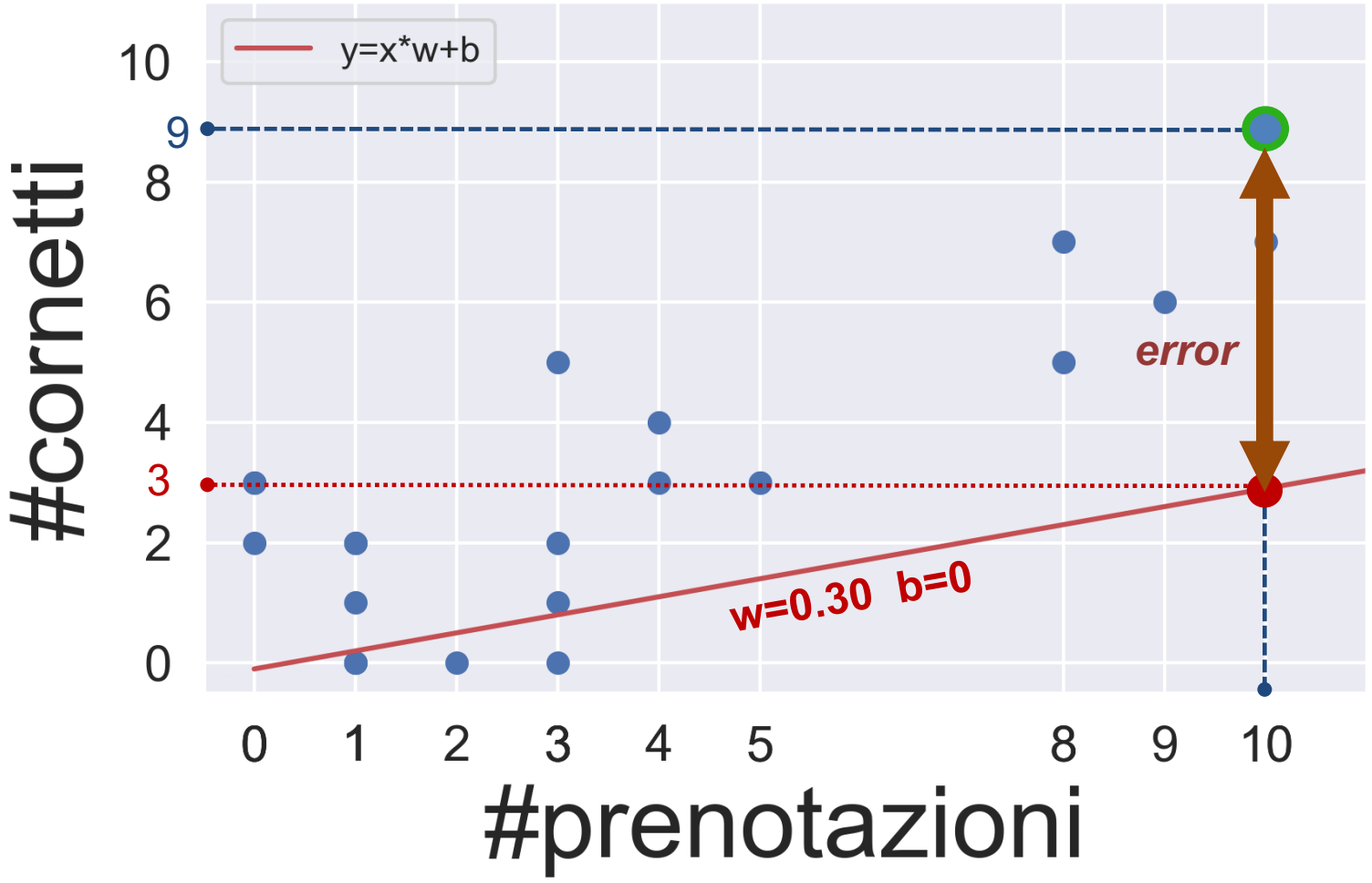


Seconda iterazione
w= 0.30
b =0

```
def loss(X,Y,w,b):
    error=prediction(X,w,b) - Y
    squared_error = error**2
```

OBIETTIVO:
trovare i valori di **w** e **b** della retta che **minimizzano** la **loss**

STRATEGIA PER TROVARE LA «BEST LINE» ?



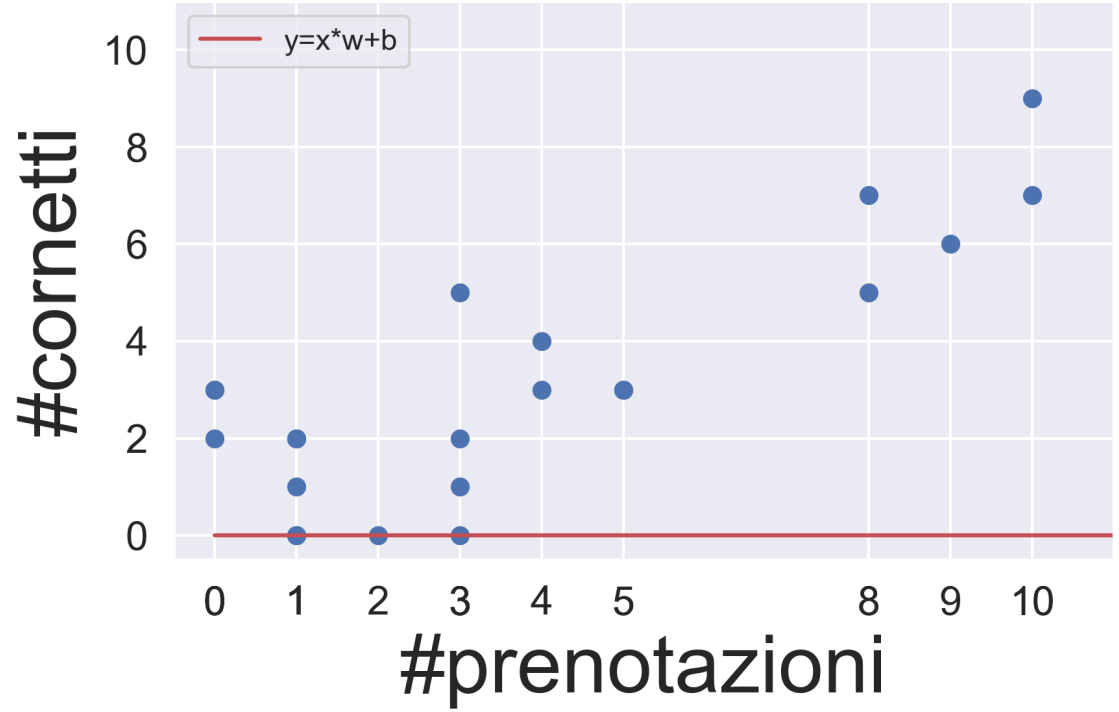
Seconda iterazione
w= 0.30
b =0

```
def loss(X,Y,w,b):
    error=prediction(X,w,b) - Y
    squared_error = error**2
    return np.average(squared_error)
```

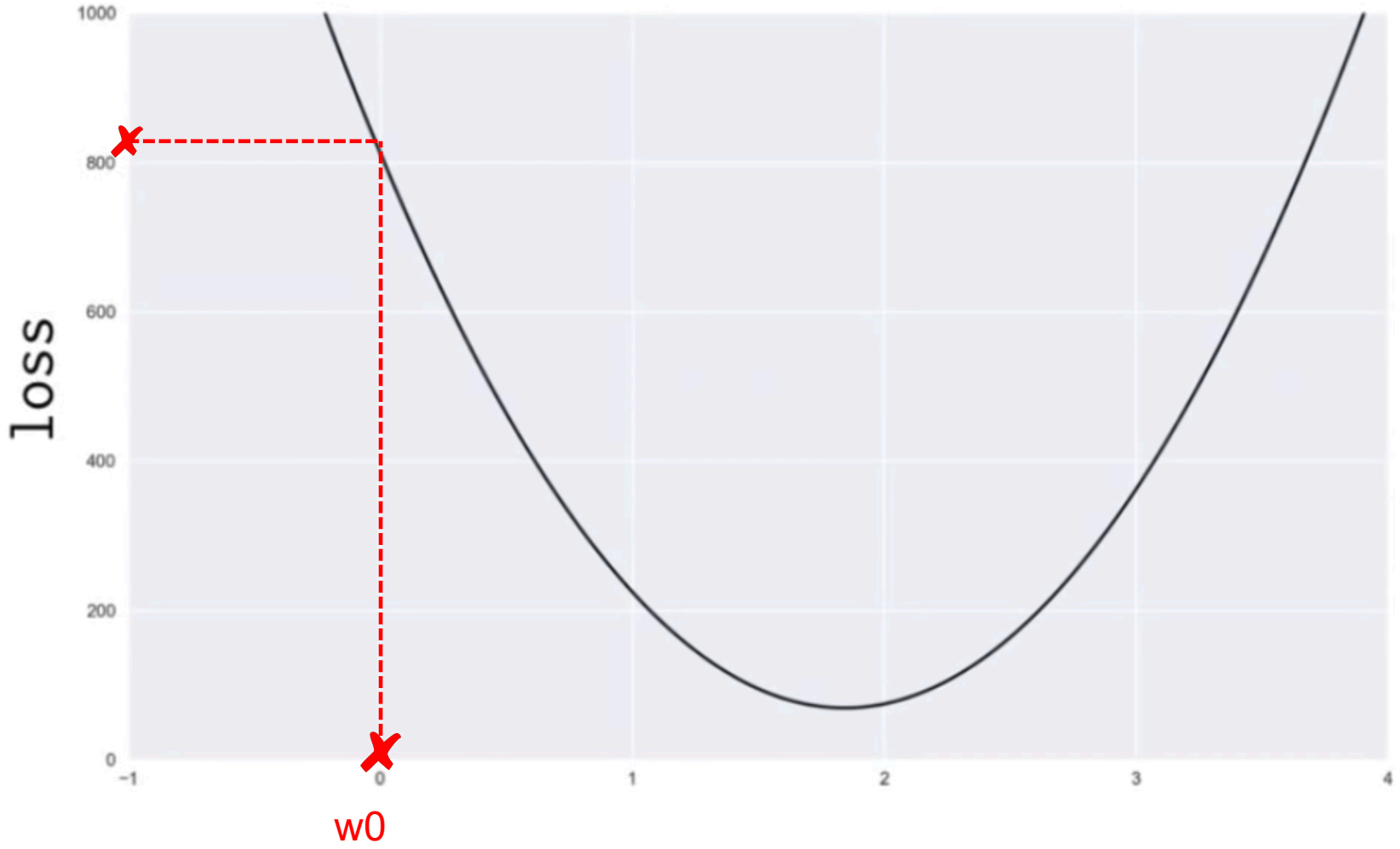
loss=400

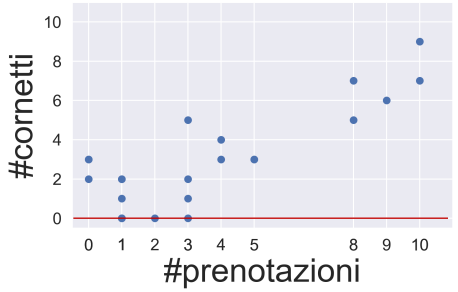
OBIETTIVO:
trovare i valori di **w** e **b** della retta che **minimizzano** la **loss**

ITERO la ricerca della «minima loss» per #N volte

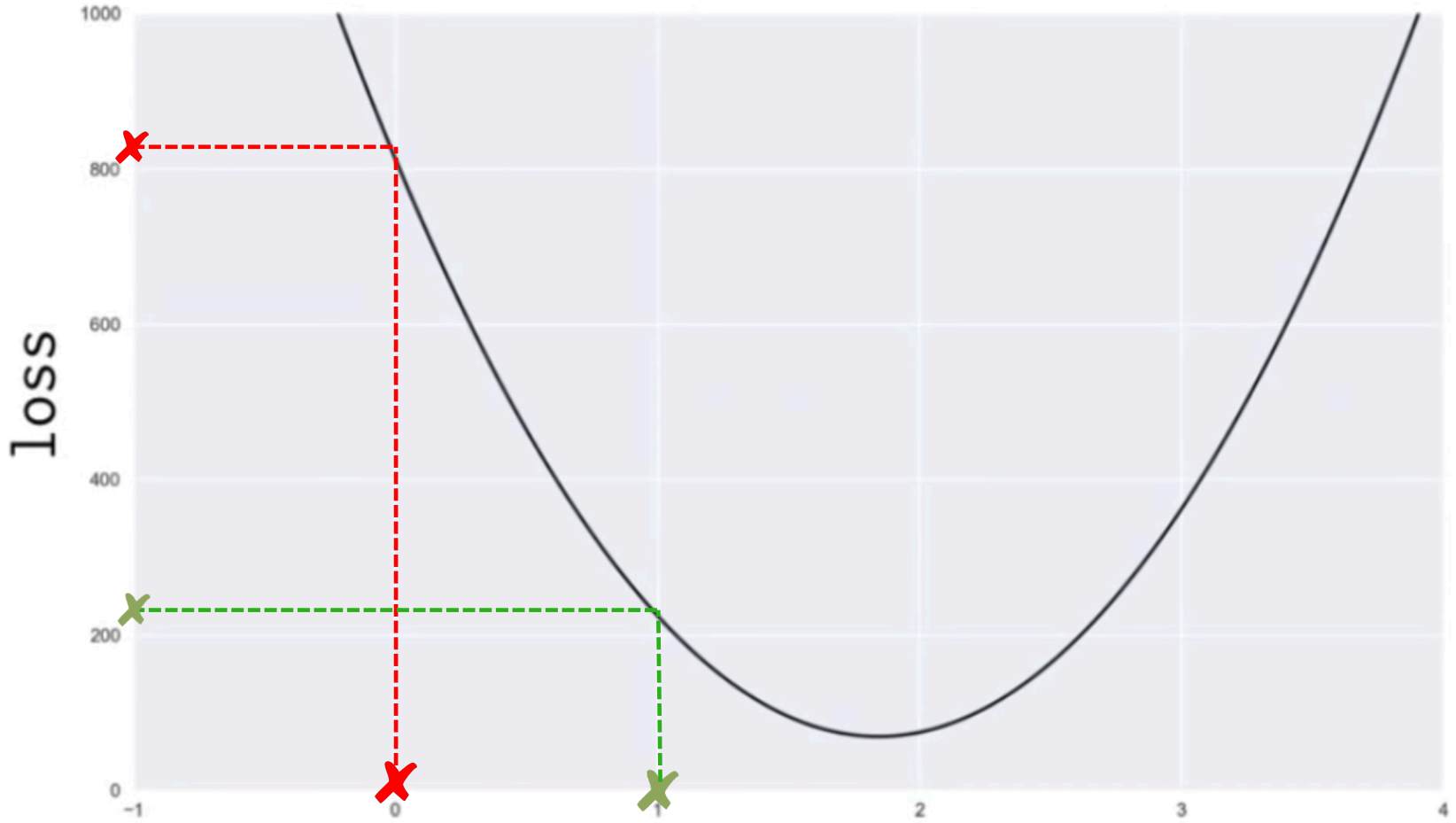


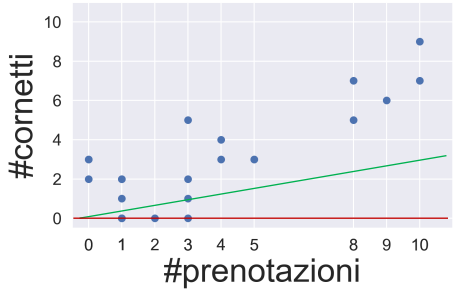
DISEGNIAMO LA FUNZIONE DI LOSS



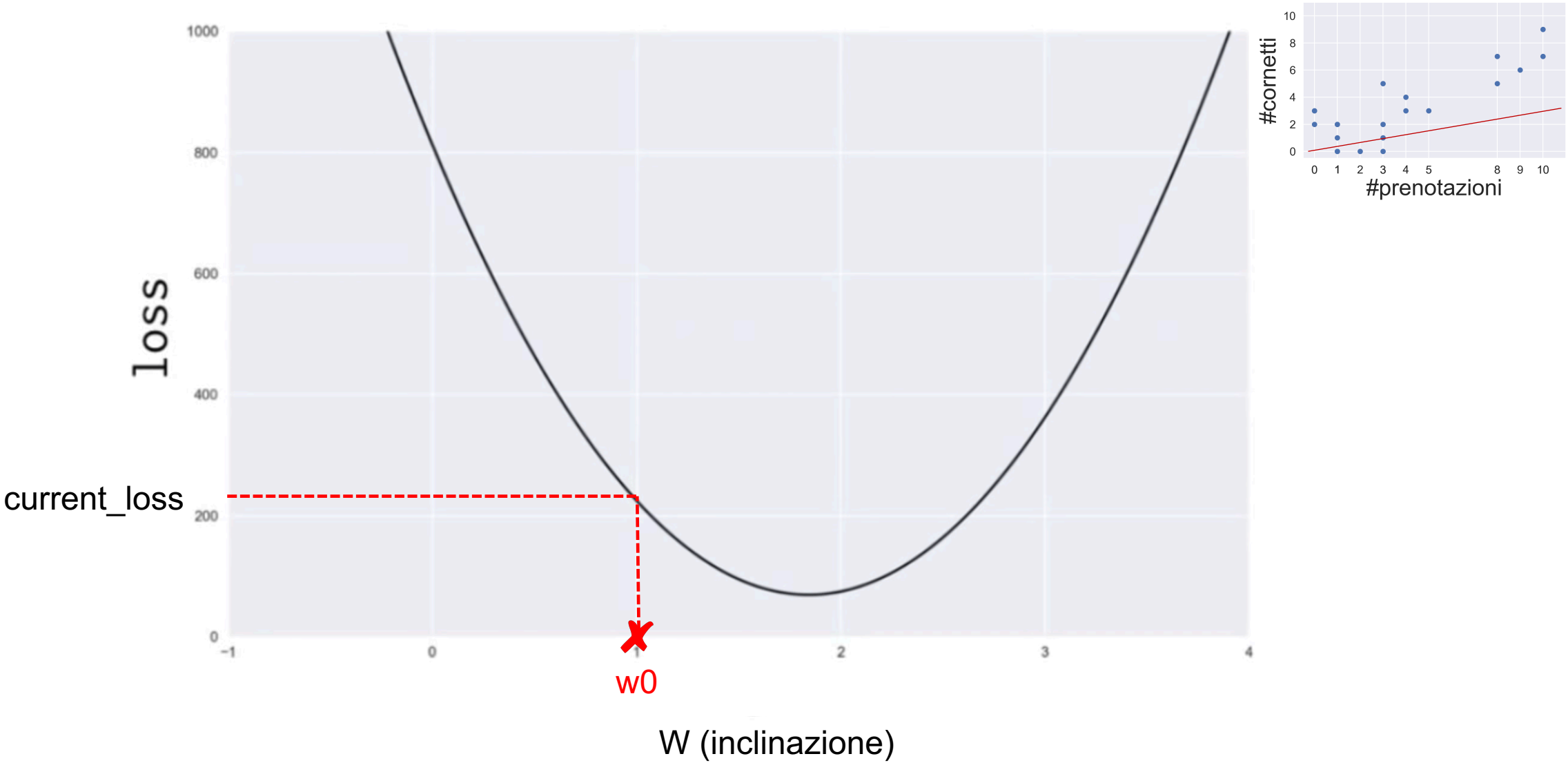


DISEGNIAMO LA FUNZIONE DI LOSS

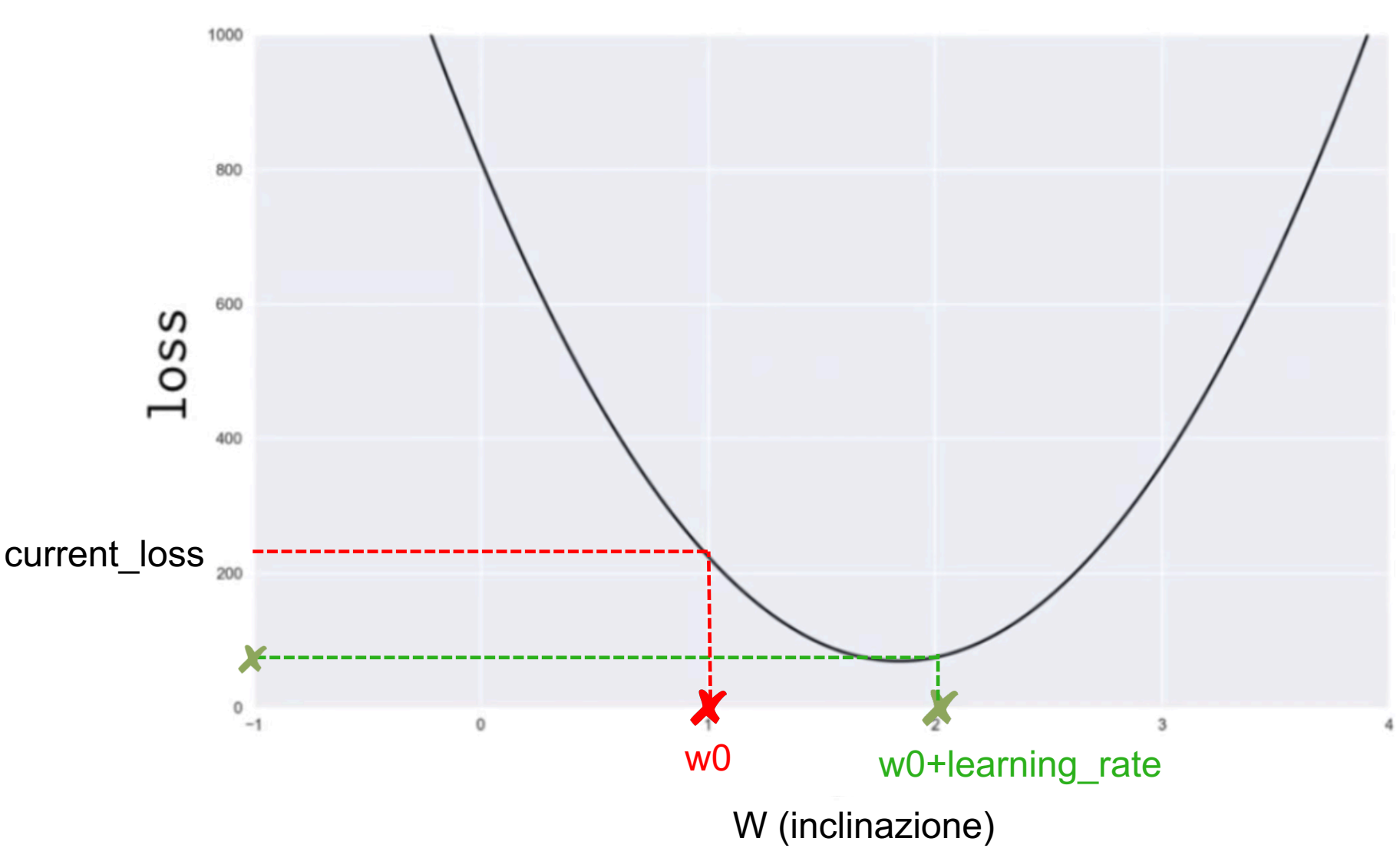


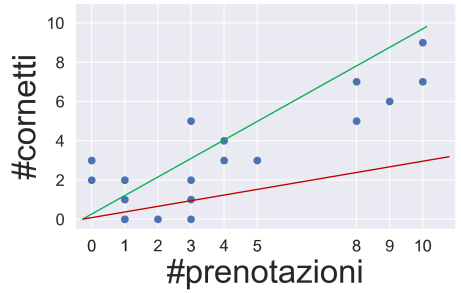


DISEGNIAMO LA FUNZIONE DI LOSS

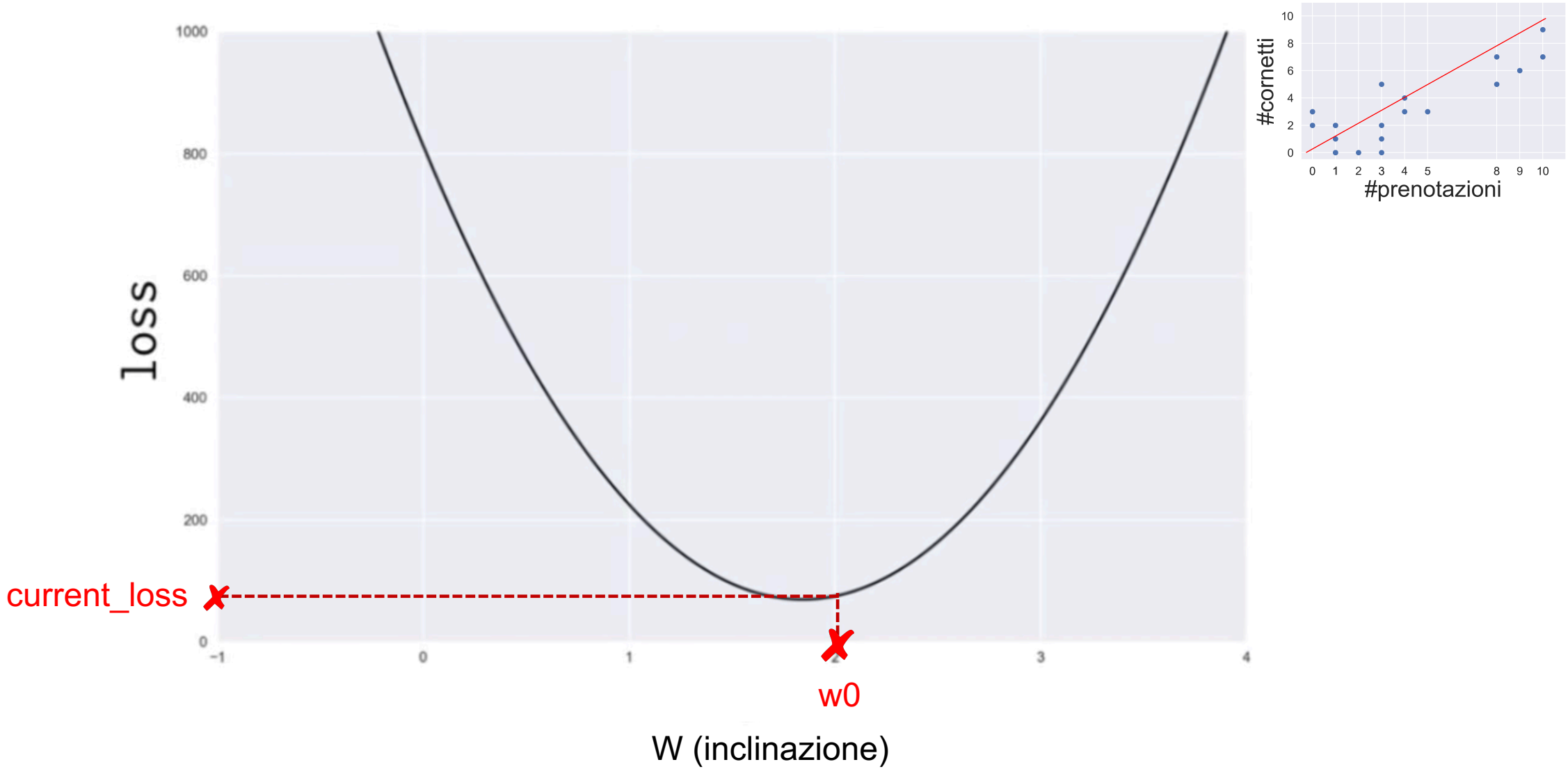


DISEGNIAMO LA FUNZIONE DI LOSS

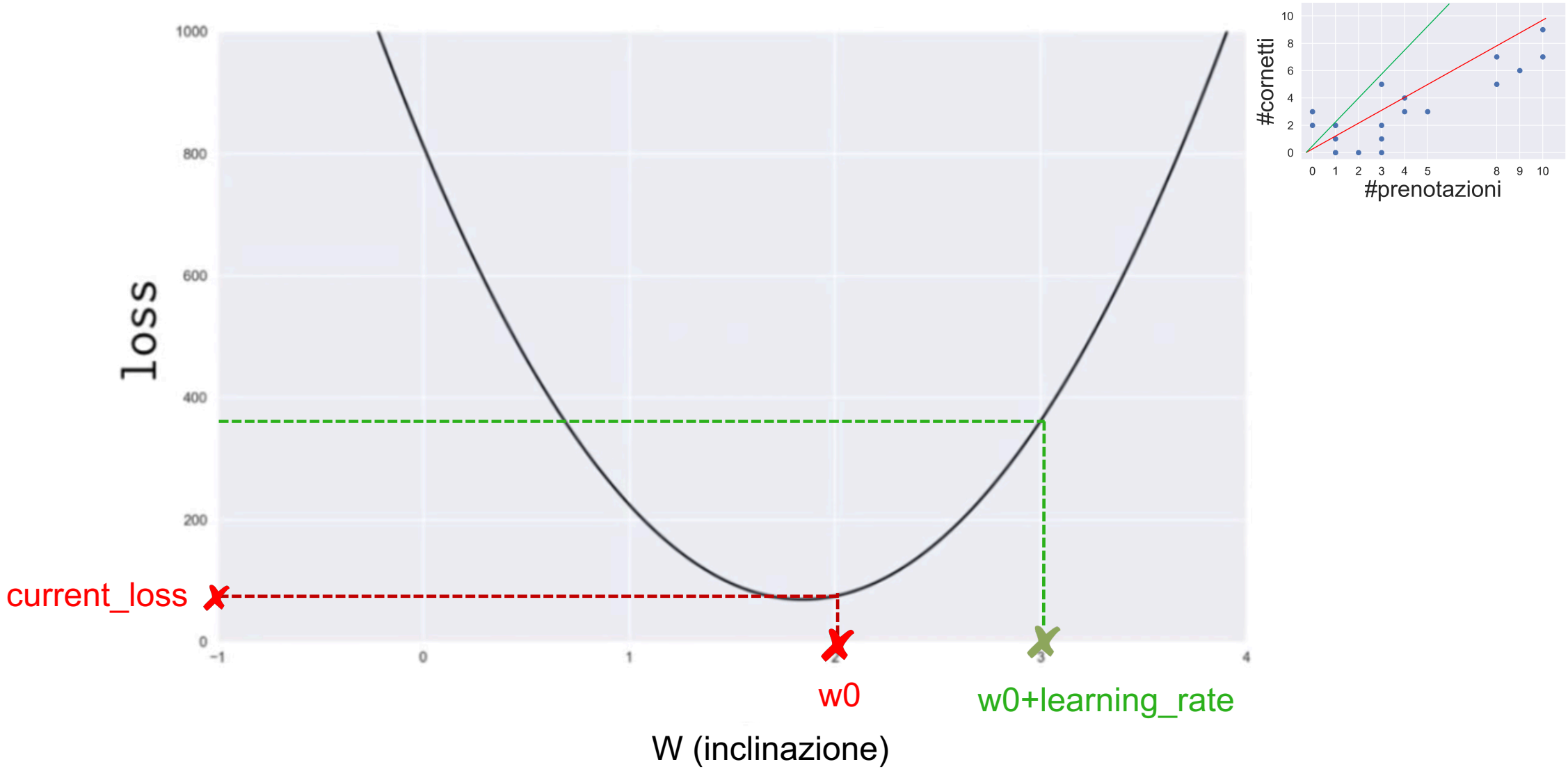




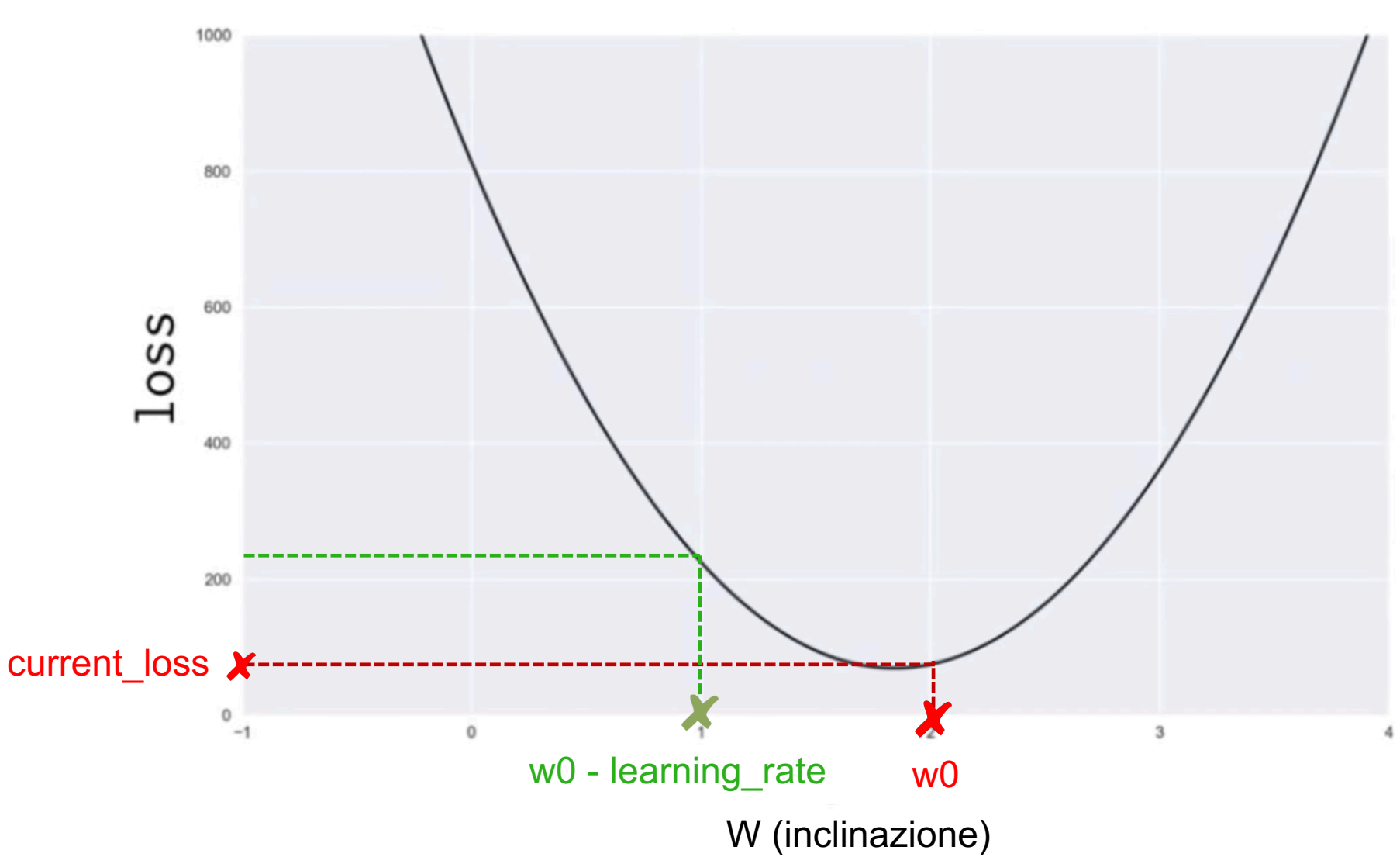
DISEGNIAMO LA FUNZIONE DI LOSS

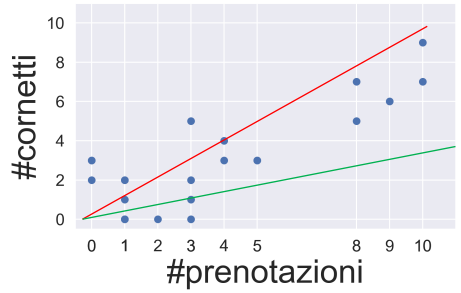


DISEGNIAMO LA FUNZIONE DI LOSS

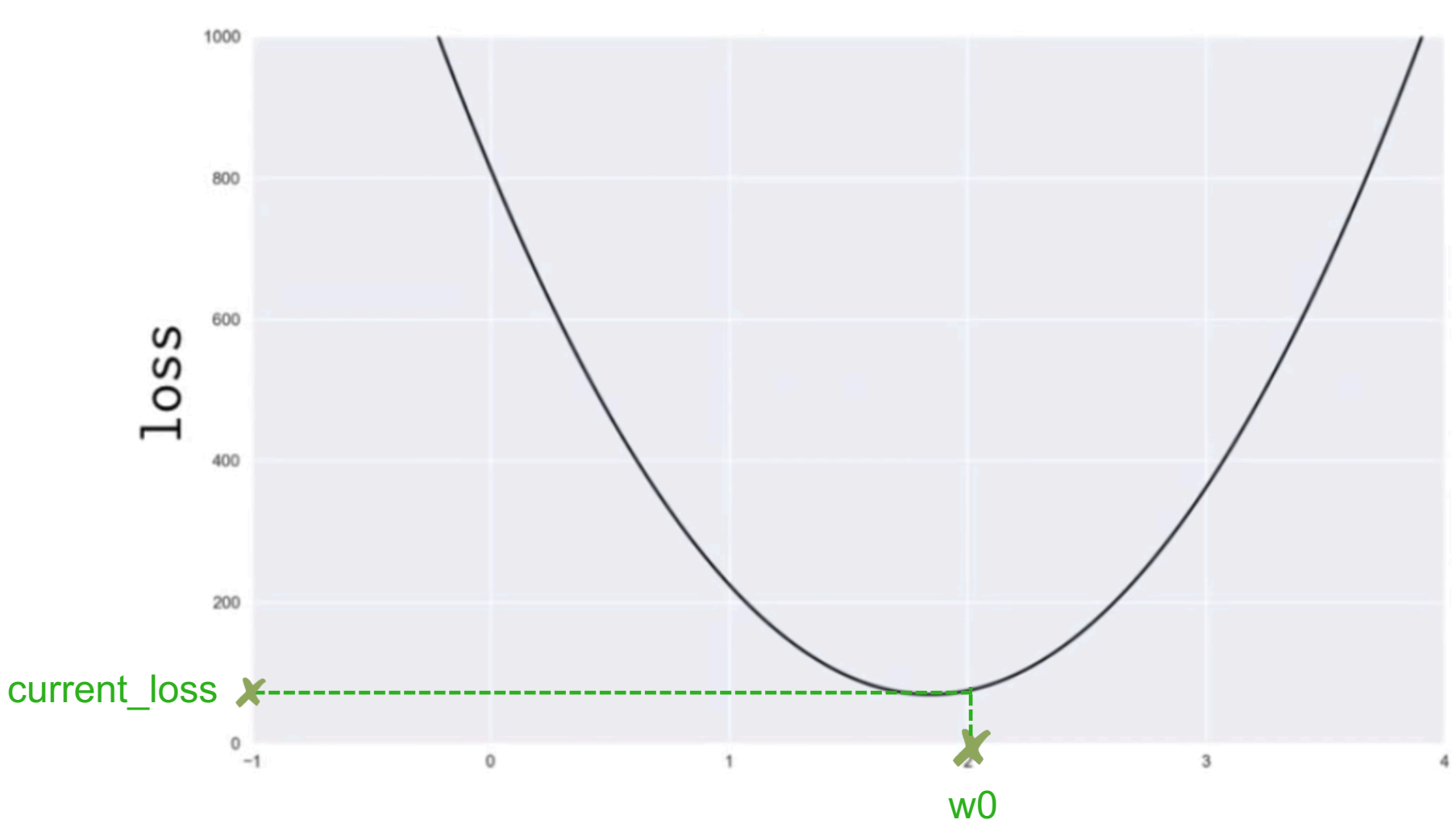


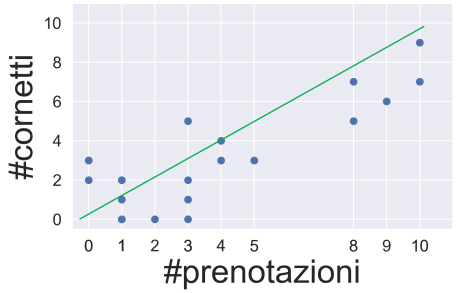
DISEGNIAMO LA FUNZIONE DI LOSS



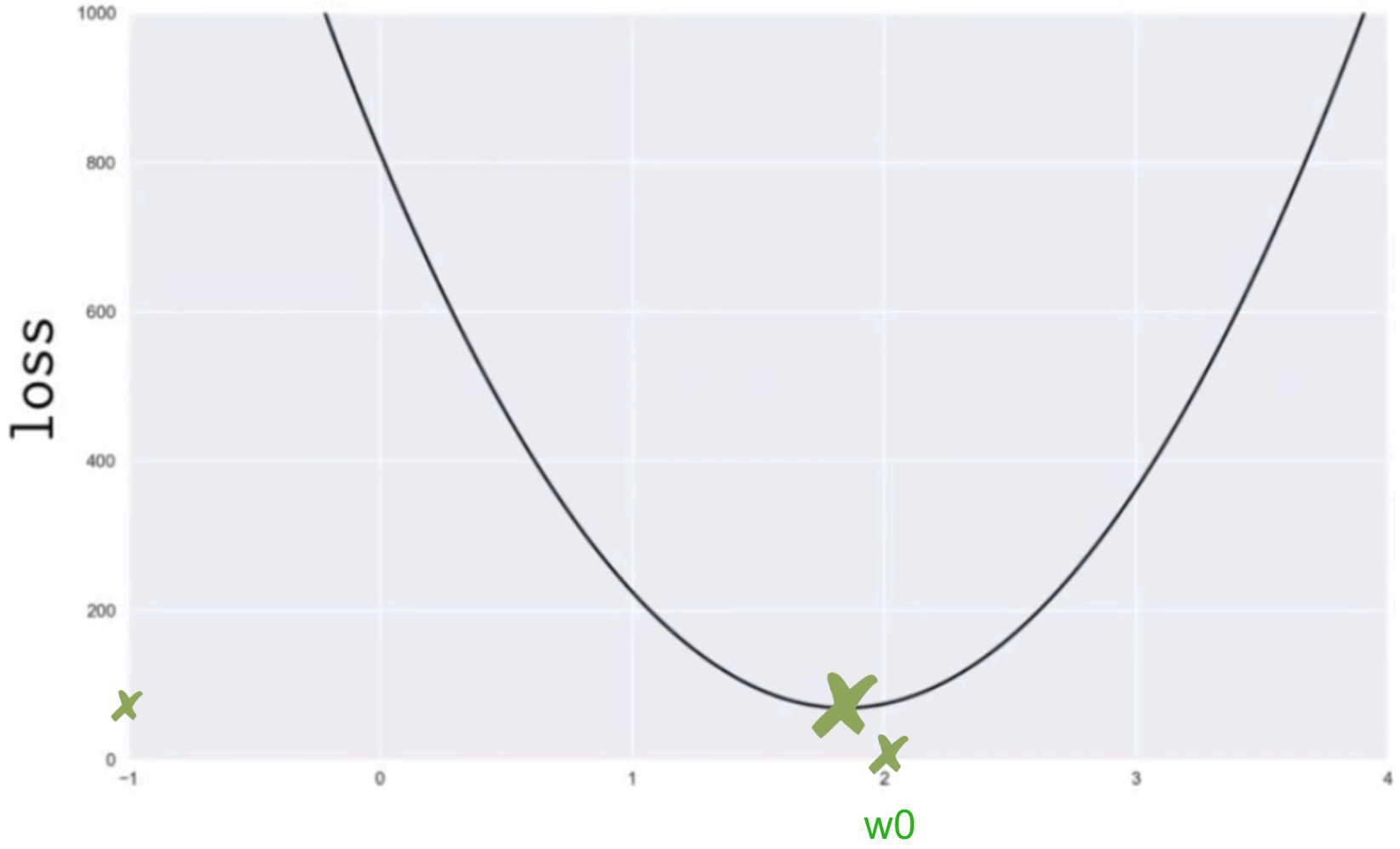


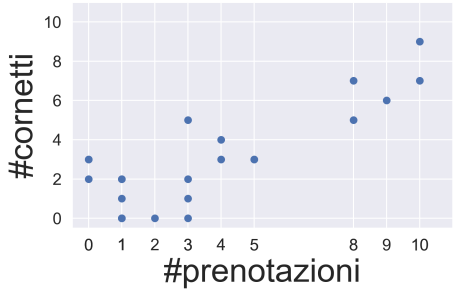
DISEGNIAMO LA FUNZIONE DI LOSS





DISEGNIAMO LA FUNZIONE DI LOSS





Implementiamo la fase di training:

```
def training(X,Y,iteration,learning_rate):
```

Scriviamo la funzione di training:

```
def training(X,Y,iteration,learning_rate):
```

Scriviamo la funzione di training:

```
def training(X,Y,iteration,learning_rate):
    w,b =0,0
```

Scriviamo la funzione di training:

```
def training(X,Y,iteration,learning_rate):
    w,b =0,0
    for i in range(iteration):
```

Scriviamo la funzione di training:

```
def training(X,Y,iteration,learning_rate):
    w,b =0,0
    for i in range(iteration):
        current_loss = loss(X,Y,w,b)
```

Scriviamo la funzione di training:

```
def training(X,Y,iteration,learning_rate):
    w,b =0,0
    for i in range(iteration):
        current_loss = loss(X,Y,w,b)
        if(loss(X,Y,w+learning_rate,b)<current_loss): #aumento l'inclinazione
```

Scriviamo la funzione di training:

```
def training(X,Y,iteration,learning_rate):
    w,b =0,0
    for i in range(iteration):
        current_loss = loss(X,Y,w,b)
        if(loss(X,Y,w+learning_rate,b)<current_loss): #aumento l'inclinazione
            w+=learning_rate
```

Scriviamo la funzione di training:

```
def training(X,Y,iteration,learning_rate):
    w,b =0,0
    for i in range(iteration):
        current_loss = loss(X,Y,w,b)
        if(loss(X,Y,w+learning_rate,b)<current_loss): #aumento l'inclinazione
            w+=learning_rate
        elif(loss(X,Y,w-learning_rate,b)<current_loss): #diminuisco l'inclinazione
```

Scriviamo la funzione di training:

```
def training(X,Y,iteration,learning_rate):
    w,b =0,0
    for i in range(iteration):
        current_loss = loss(X,Y,w,b)
        if(loss(X,Y,w+learning_rate,b)<current_loss): #aumento l'inclinazione
            w+=learning_rate
        elif(loss(X,Y,w-learning_rate,b)<current_loss): #diminuisco l'inclinazione
            w-=learning_rate
```

Scriviamo la funzione di training:

```
def training(X,Y,iteration,learning_rate):
    w,b =0,0
    for i in range(iteration):
        current_loss = loss(X,Y,w,b)
        if(loss(X,Y,w+learning_rate,b)<current_loss): #aumento l'inclinazione
            w+=learning_rate
        elif(loss(X,Y,w-learning_rate,b)<current_loss): #diminuisco l'inclinazione
            w-=learning_rate
        elif(loss(X,Y,w,b+learning_rate)<current_loss): #traslo la retta più SU
```

Scriviamo la funzione di training:

```
def training(X,Y,iteration,learning_rate):
    w,b =0,0
    for i in range(iteration):
        current_loss = loss(X,Y,w,b)
        if(loss(X,Y,w+learning_rate,b)<current_loss): #aumento l'inclinazione
            w+=learning_rate
        elif(loss(X,Y,w-learning_rate,b)<current_loss): #diminuisco l'inclinazione
            w-=learning_rate
        elif(loss(X,Y,w,b+learning_rate)<current_loss): #traslo la retta più SU
            b+=learning_rate
```

Scriviamo la funzione di training:

```
def training(X,Y,iteration,learning_rate):
    w,b =0,0
    for i in range(iteration):
        current_loss = loss(X,Y,w,b)
        if(loss(X,Y,w+learning_rate,b)<current_loss): #aumento l'inclinazione
            w+=learning_rate
        elif(loss(X,Y,w-learning_rate,b)<current_loss): #diminuisco l'inclinazione
            w-=learning_rate
        elif(loss(X,Y,w,b+learning_rate)<current_loss): #traslo la retta più SU
            b+=learning_rate
        elif(loss(X,Y,w,b-learning_rate)<current_loss): #traslo la retta più GIU'
```

Scriviamo la funzione di training:

```
def training(X,Y,iteration,learning_rate):
    w,b =0,0
    for i in range(iteration):
        current_loss = loss(X,Y,w,b)
        if(loss(X,Y,w+learning_rate,b)<current_loss): #aumento l'inclinazione
            w+=learning_rate
        elif(loss(X,Y,w-learning_rate,b)<current_loss): #diminuisco l'inclinazione
            w-=learning_rate
        elif(loss(X,Y,w,b+learning_rate)<current_loss): #traslo la retta più SU
            b+=learning_rate
        elif(loss(X,Y,w,b-learning_rate)<current_loss): #traslo la retta più GIU'
            b-=learning_rate
```

Scriviamo la funzione di training:

```
def training(X,Y,iteration,learning_rate):
    w,b =0,0
    for i in range(iteration):
        current_loss = loss(X,Y,w,b)
        if(loss(X,Y,w+learning_rate,b)<current_loss): #aumento l'inclinazione
            w+=learning_rate
        elif(loss(X,Y,w-learning_rate,b)<current_loss): #diminuisco l'inclinazione
            w-=learning_rate
        elif(loss(X,Y,w,b+learning_rate)<current_loss): #traslo la retta più SU
            b+=learning_rate
        elif(loss(X,Y,w,b-learning_rate)<current_loss): #traslo la retta più GIU'
            b-=learning_rate
        else:
```

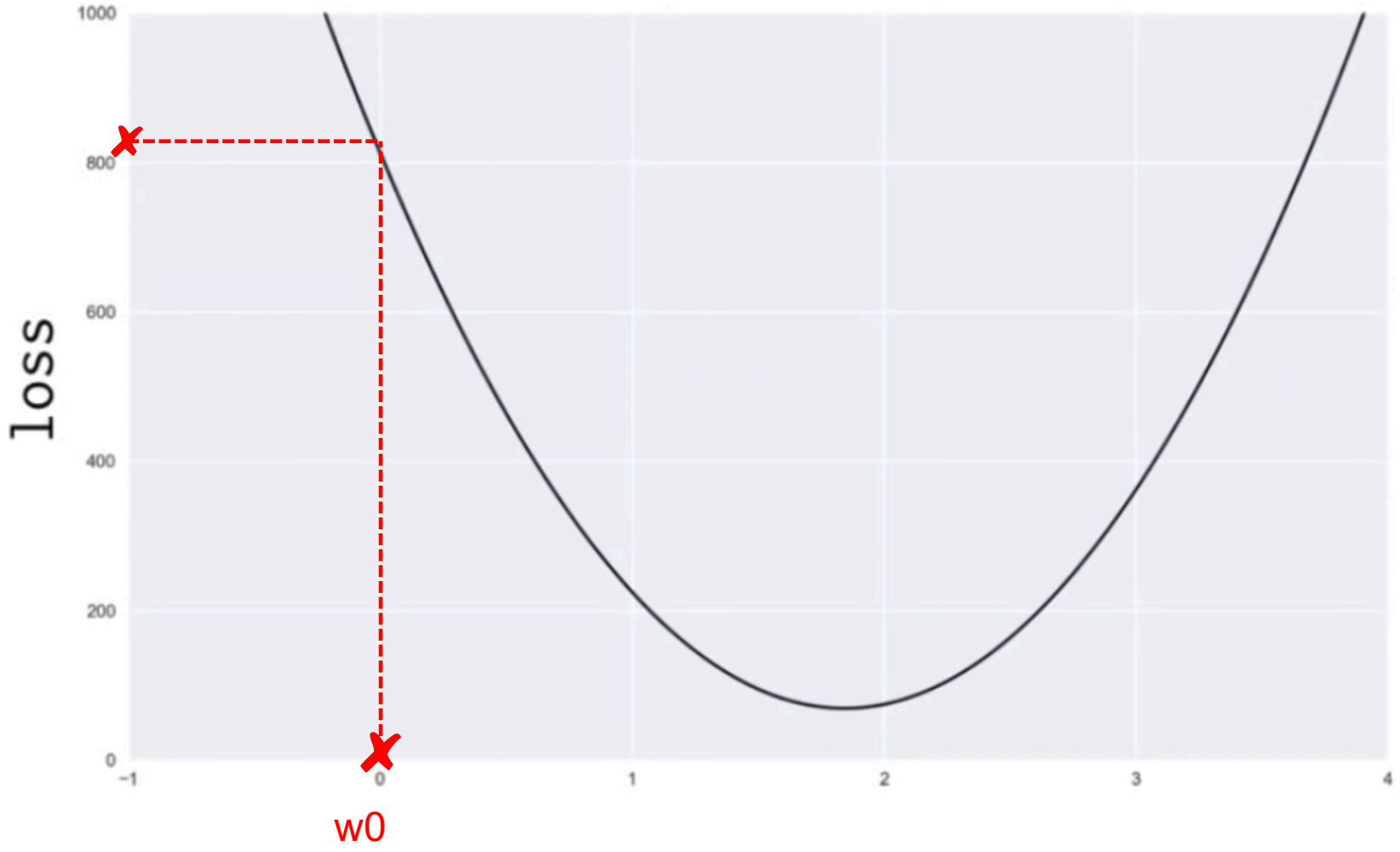
Scriviamo la funzione di training:

```
def training(X,Y,iteration,learning_rate):
    w,b =0,0
    for i in range(iteration):
        current_loss = loss(X,Y,w,b)
        if(loss(X,Y,w+learning_rate,b)<current_loss): #aumento l'inclinazione
            w+=learning_rate
        elif(loss(X,Y,w-learning_rate,b)<current_loss): #diminuisco l'inclinazione
            w-=learning_rate
        elif(loss(X,Y,w,b+learning_rate)<current_loss): #traslo la retta più SU
            b+=learning_rate
        elif(loss(X,Y,w,b-learning_rate)<current_loss): #traslo la retta più GIU'
            b-=learning_rate
        else:
            return w,b
```

Scriviamo la funzione di training:

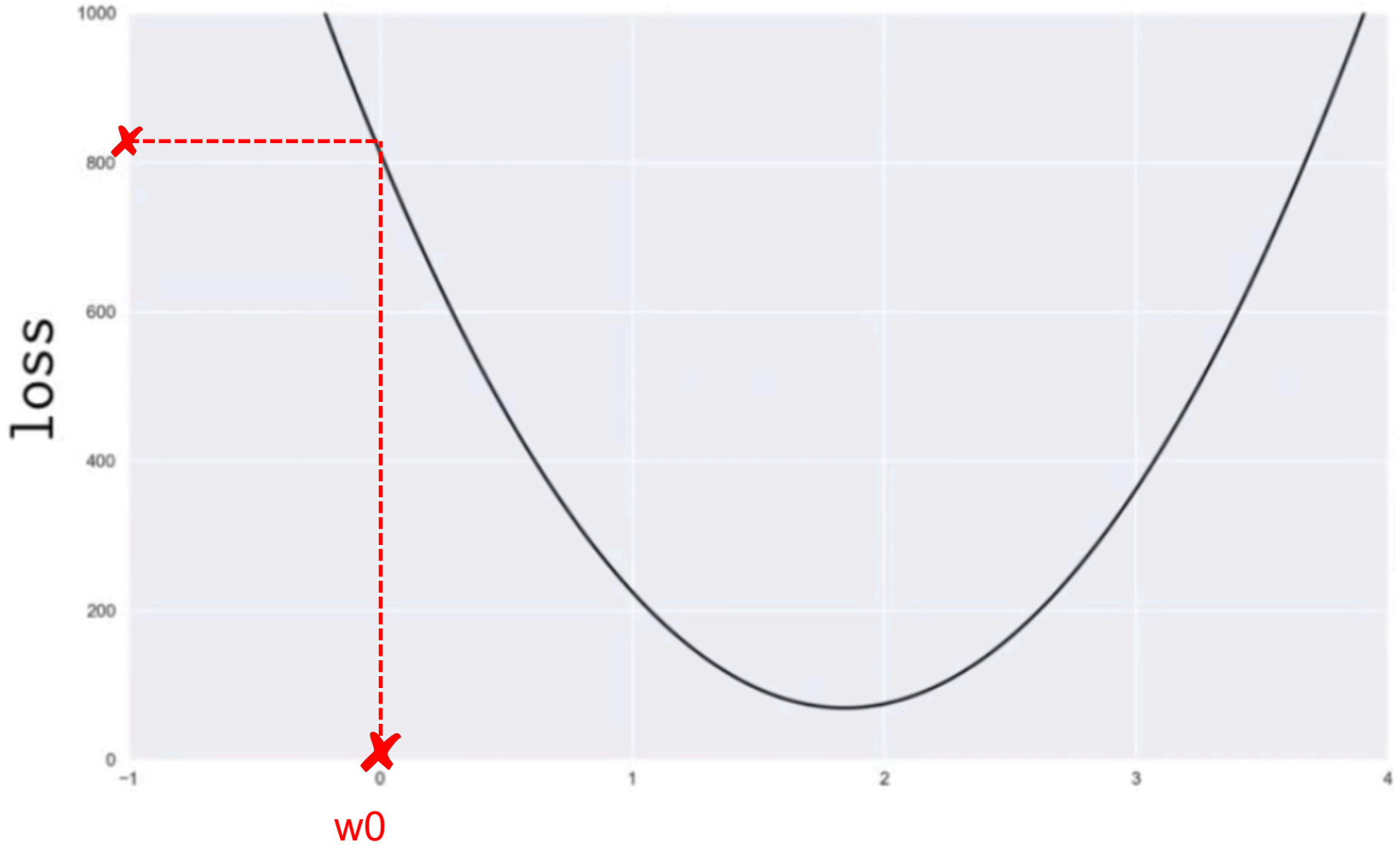
```
def training(X,Y,iteration,learning_rate):
    w,b =0,0
    for i in range(iteration):
        current_loss = loss(X,Y,w,b)
        if(loss(X,Y,w+learning_rate,b)<current_loss): #aumento l'inclinazione
            w+=learning_rate
        elif(loss(X,Y,w-learning_rate,b)<current_loss): #diminuisco l'inclinazione
            w-=learning_rate
        elif(loss(X,Y,w,b+learning_rate)<current_loss): #traslo la retta più SU
            b+=learning_rate
        elif(loss(X,Y,w,b-learning_rate)<current_loss): #traslo la retta più GIU'
            b-=learning_rate
        else:
            return w,b

    raise Exception(f"Non sono riuscito a convergere con {iteration} iterazioni")
```

Funzione di loss che varia solo rispetto a w

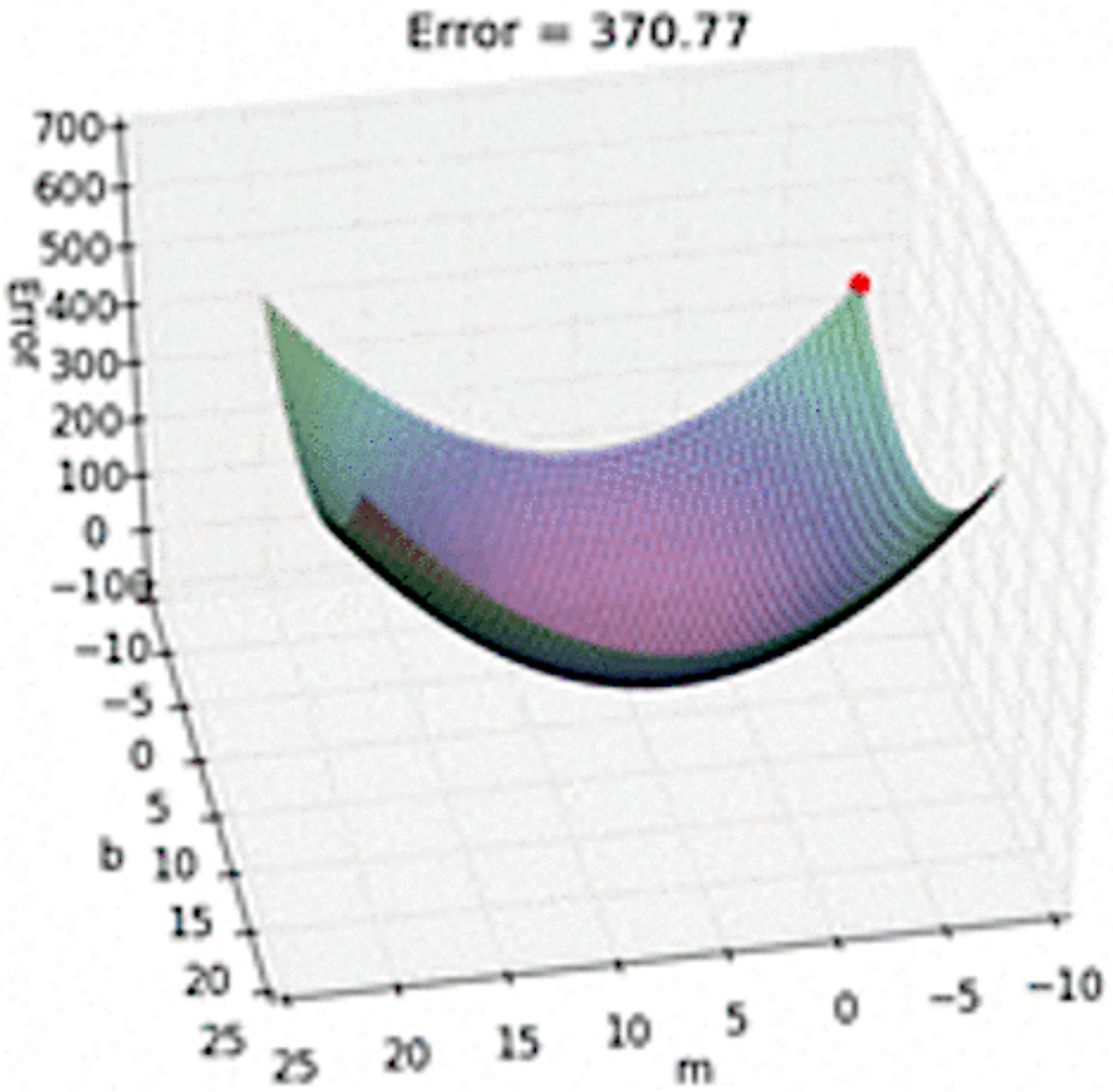


FUNZIONE DI LOSS

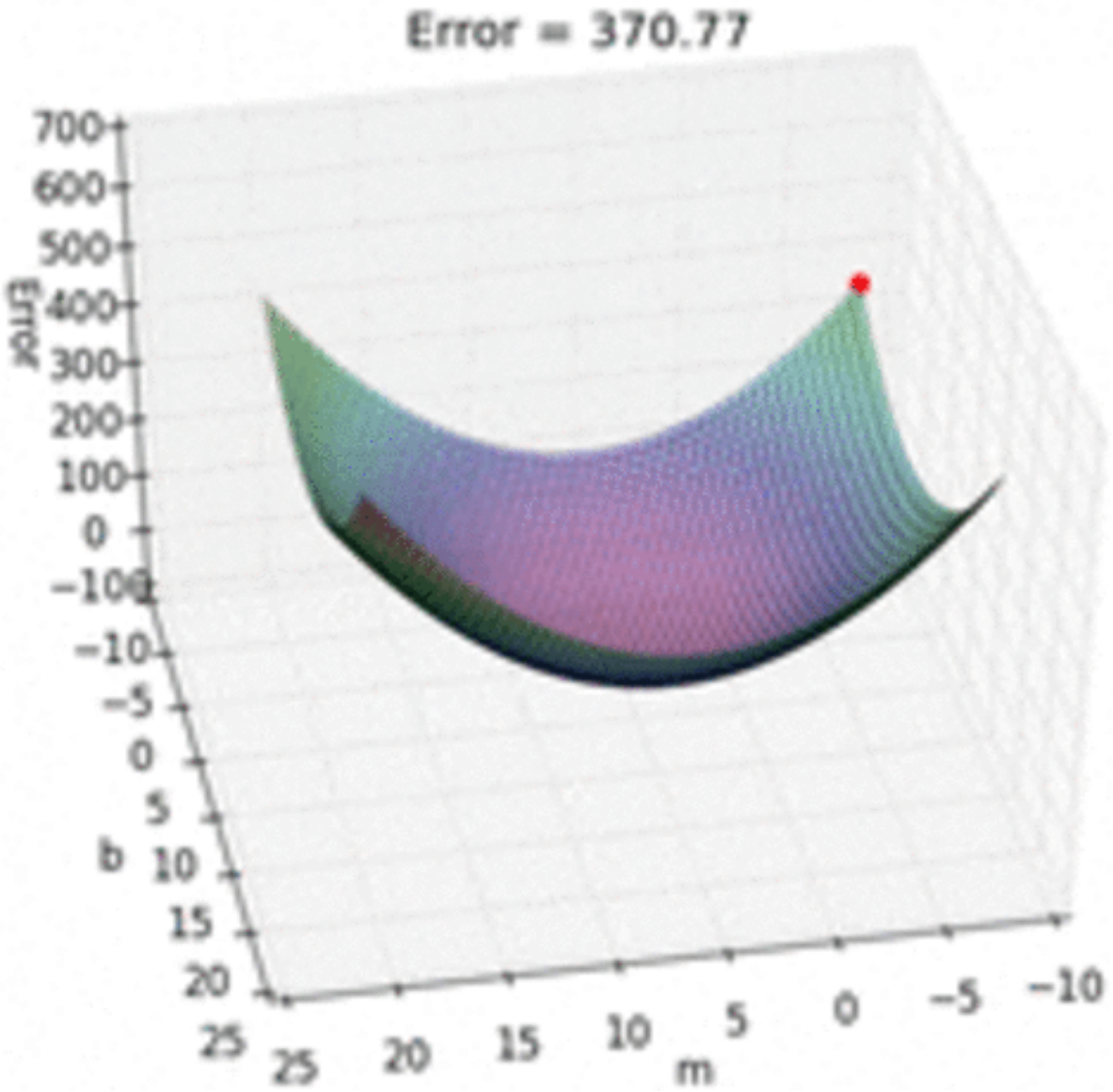
E si aggiungo b?



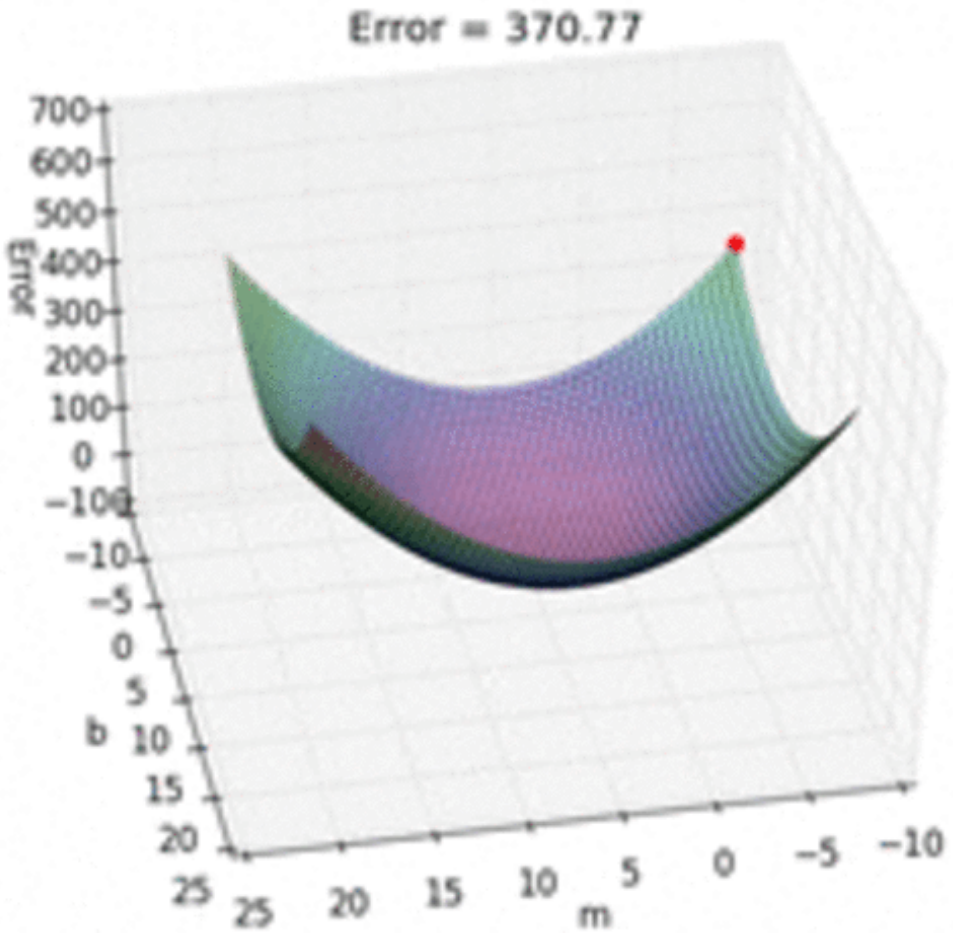
Funzione di loss rispetto a w e b

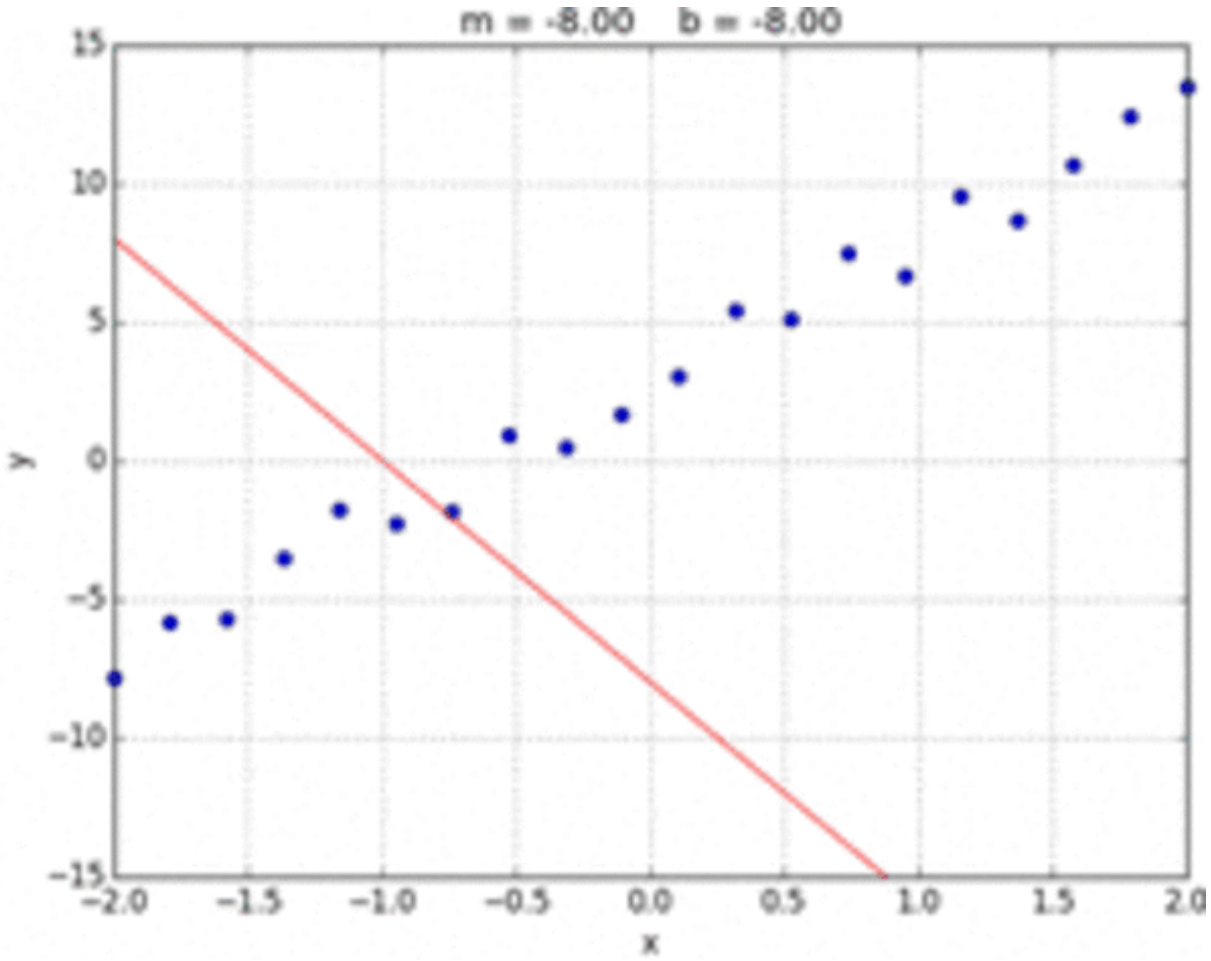


Funzione di loss rispetto a w e b



Scriviamo la funzione di training:





CODE…

http://localhost:8888/lab/tree/corso%20python/2day/01_ex/01_regression_linear.ipynb

RECAP

Apprendimento supervisionato

1. Training
2. Prediction

Se applichiamo il nostro algoritmo «regressione lineare» a problemi più complessi non scala….

SCALABILITA’

```python
def training(X,Y,iteration,learning_rate):
    w=0
    b=0
    for i in range(iteration):
        current_loss = loss(X,Y,w,b)
        print(f"iterazione {i+1} - w0: {w:2.3f} b0: {b:2.3f} - loss:{current_loss}")
        if(loss(X,Y,w+learning_rate,b)<current_loss):
            w+=learning_rate
        elif(loss(X,Y,w-learning_rate,b)<current_loss):
            w-=learning_rate
        elif(loss(X,Y,w,b+learning_rate)<current_loss):
            b+=learning_rate
        elif(loss(X,Y,w,b-learning_rate)<current_loss):
            b-=learning_rate
        else:
            return w,b

    raise Exception(f"Non sono riuscito a convergere con {iteration} iterazioni")
```

SCALABILITA’

```
def training(X,Y,iteration,learning_rate):
    w=0
    b=0
    for i in range(iteration):
        current_loss = loss(X,Y,w,b)
        print(f"iterazione {i+1} - w0: {w:2.3f} b0: {b:2.3f} - loss:{current_loss}")
        if(loss(X,Y,w+learning_rate,b)<current_loss):
            w+=learning_rate
        elif(loss(X,Y,w-learning_rate,b)<current_loss):
            w-=learning_rate
        elif(loss(X,Y,w,b+learning_rate)<current_loss):
            b+=learning_rate
        elif(loss(X,Y,w,b-learning_rate)<current_loss):
            b-=learning_rate
        elif(loss(X,Y,w+learning_rate,b-learning_rate)<current_loss):
            w+=learning_rate
            b-=learning_rate
        elif(loss(X,Y,w-learning_rate,b-learning_rate)<current_loss):
            w-=learning_rate
            b-=learning_rate
        elif(loss(X,Y,w+learning_rate,b+learning_rate)<current_loss):
            w+=learning_rate
            b+=learning_rate
        elif(loss(X,Y,w-learning_rate,b+learning_rate)<current_loss):
            w-=learning_rate
            b+=learning_rate
        else:
            return w,b

    raise Exception(f"Non sono riuscito a convergere con {iteration} iterazioni")
```

Per ogni iterazione :

- con n = **2** *parametri* (w e b)
3**n = 9 confronti

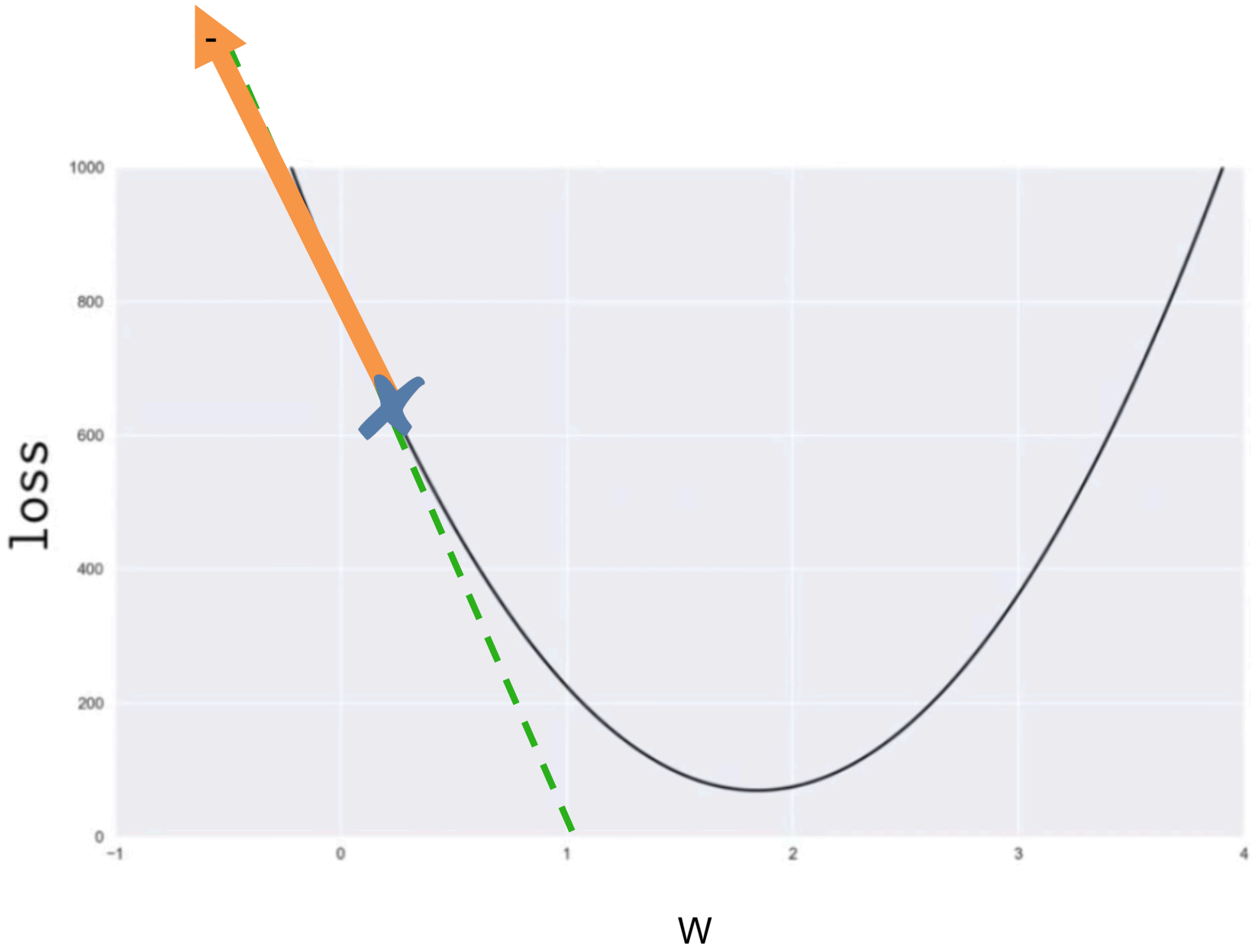
- con n = ***10*** *parametri*
3**n= 59049 confronti

- con n= **700** parametri ?

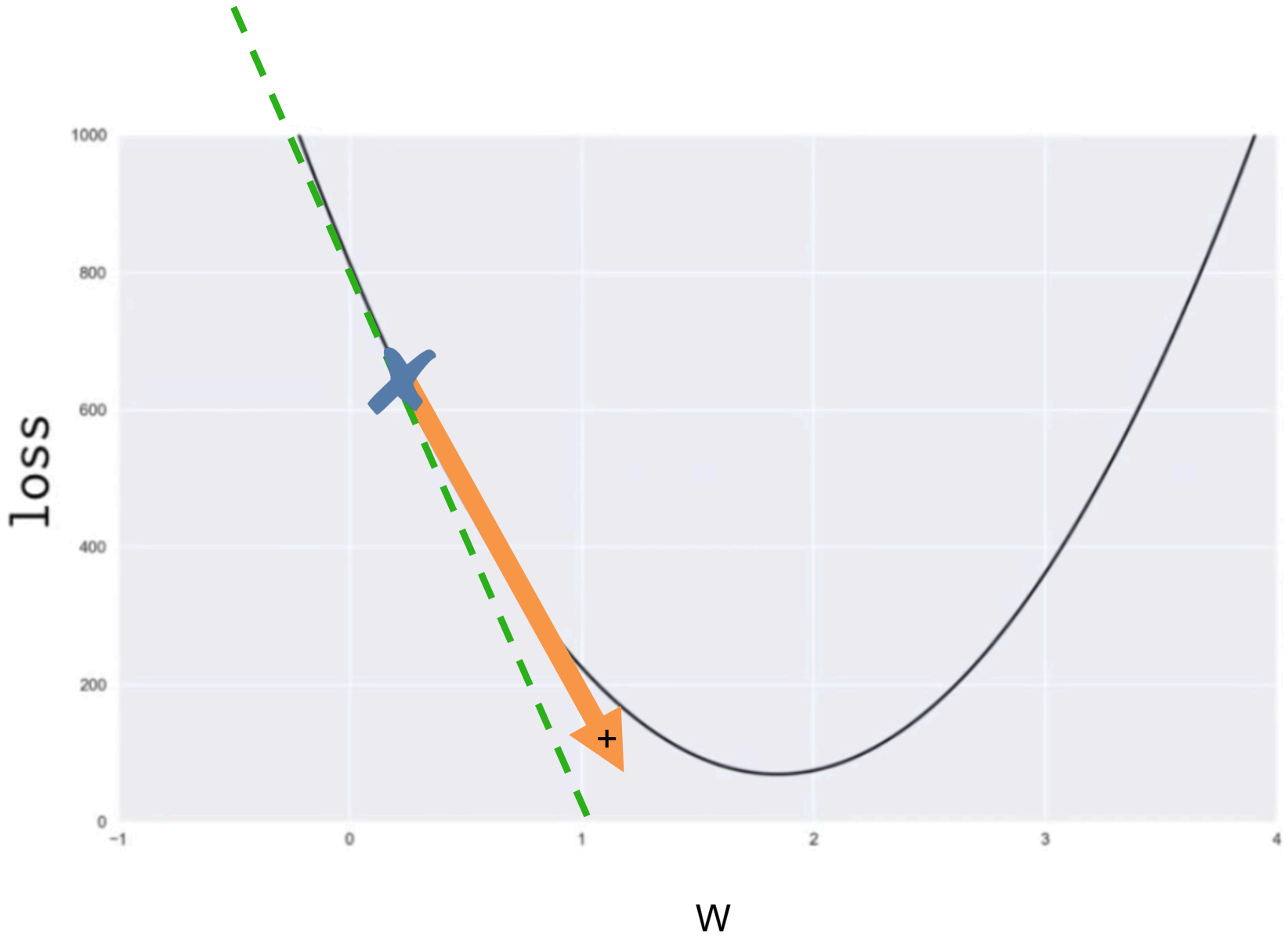
Overflow!!!

Come **scalo** la mia funzione di training?

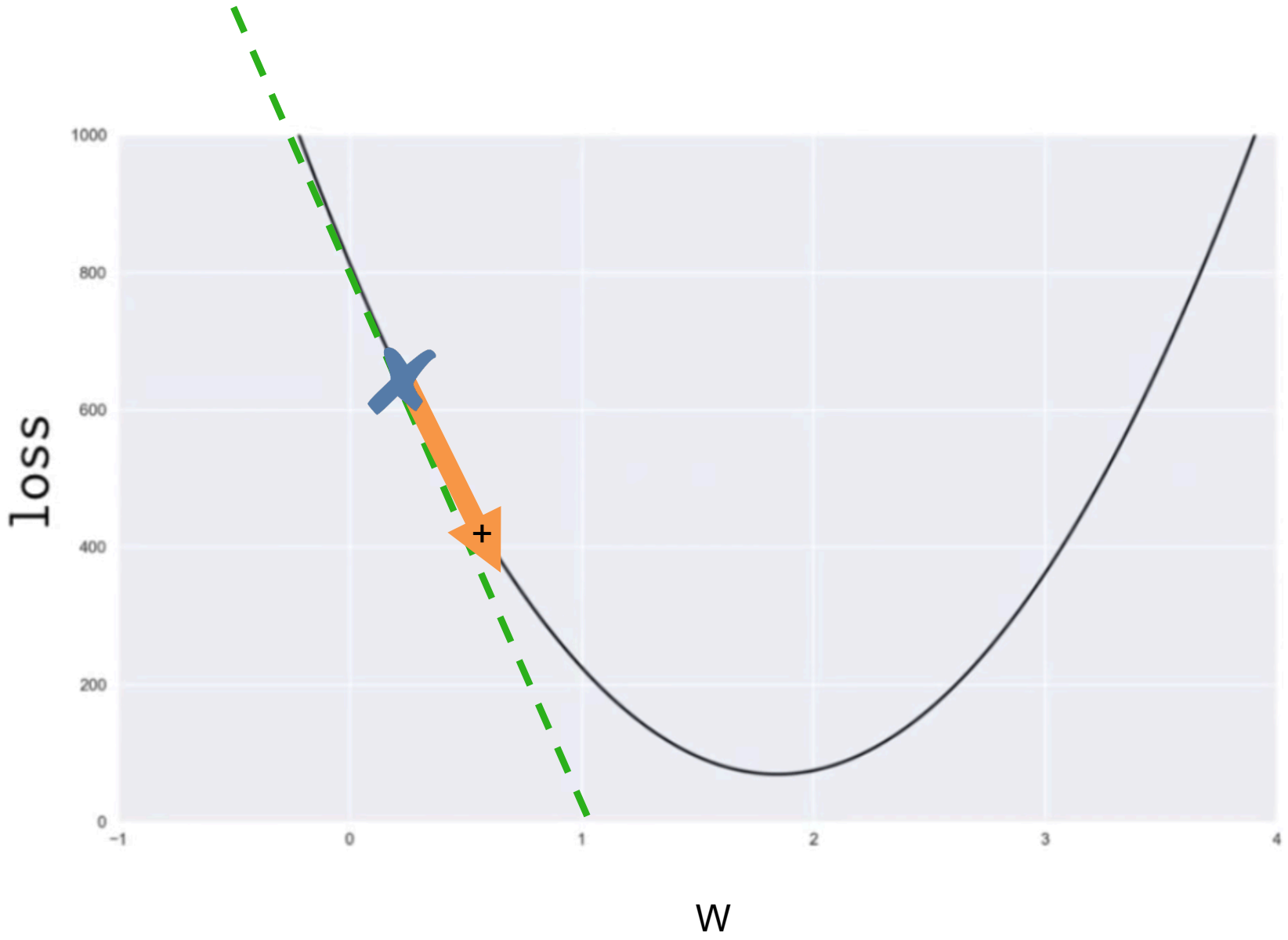
GRADIENT DESCENT



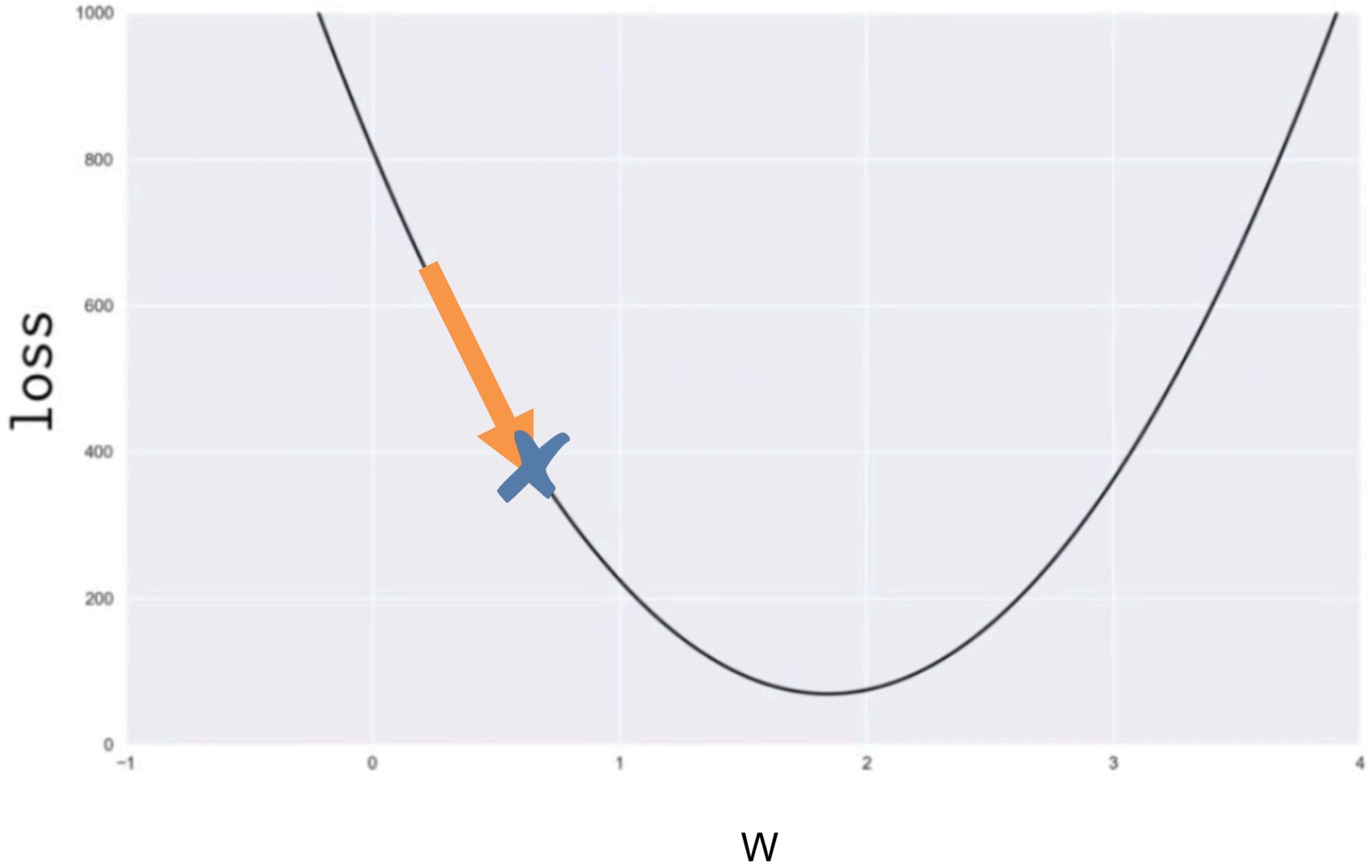
GRADIENT DESCENT



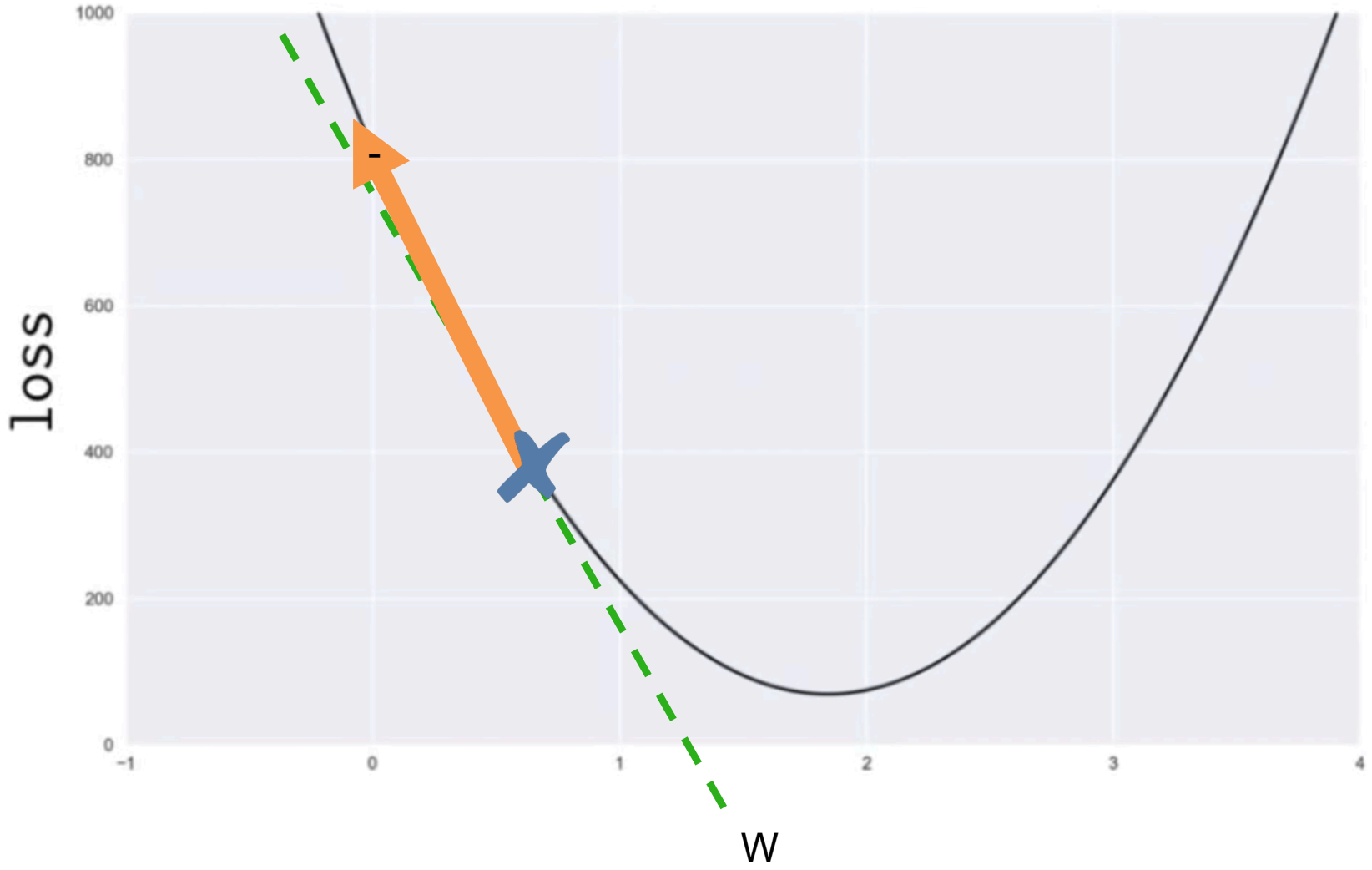
GRADIENT DESCENT



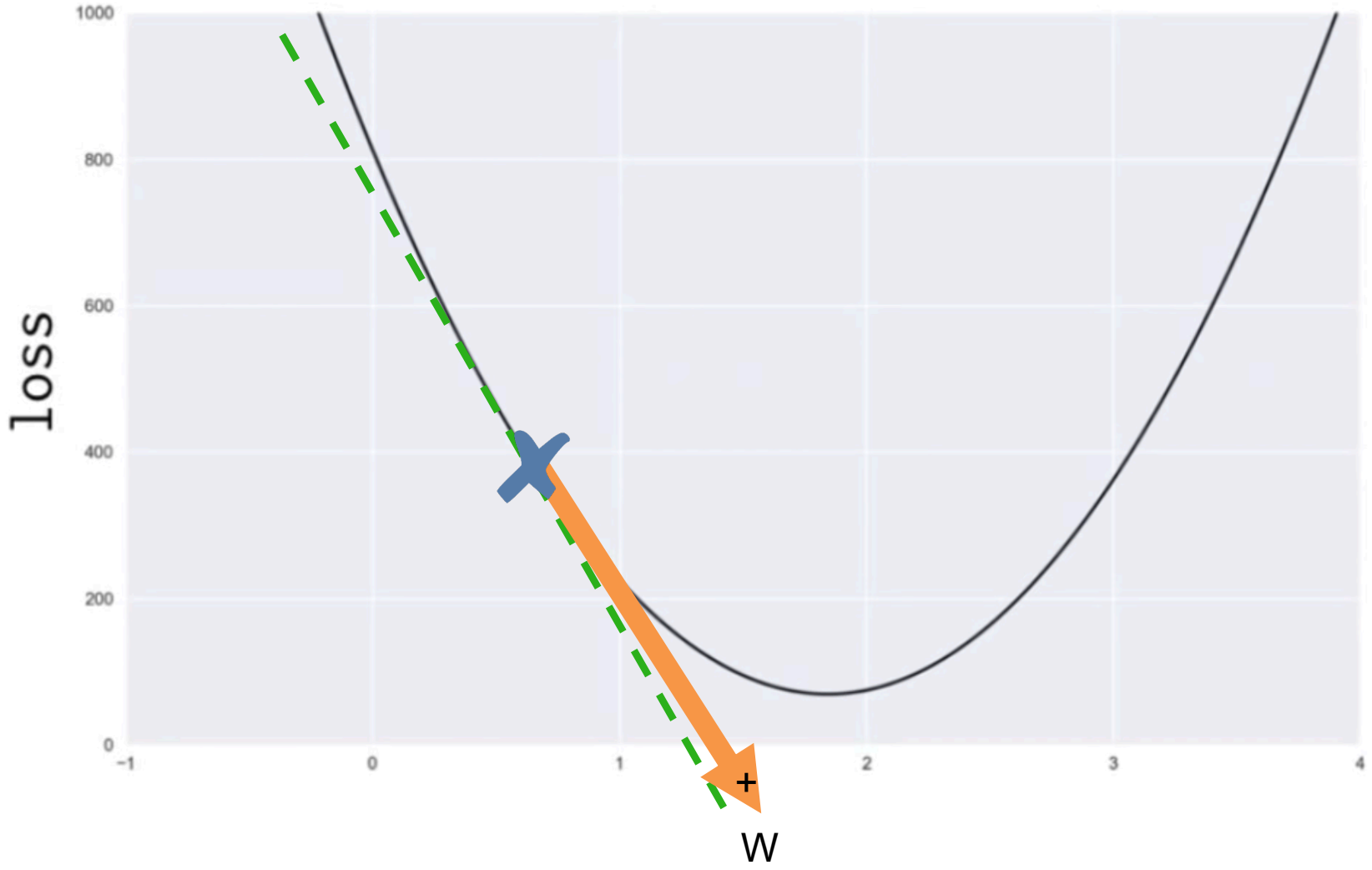
GRADIENT DESCENT



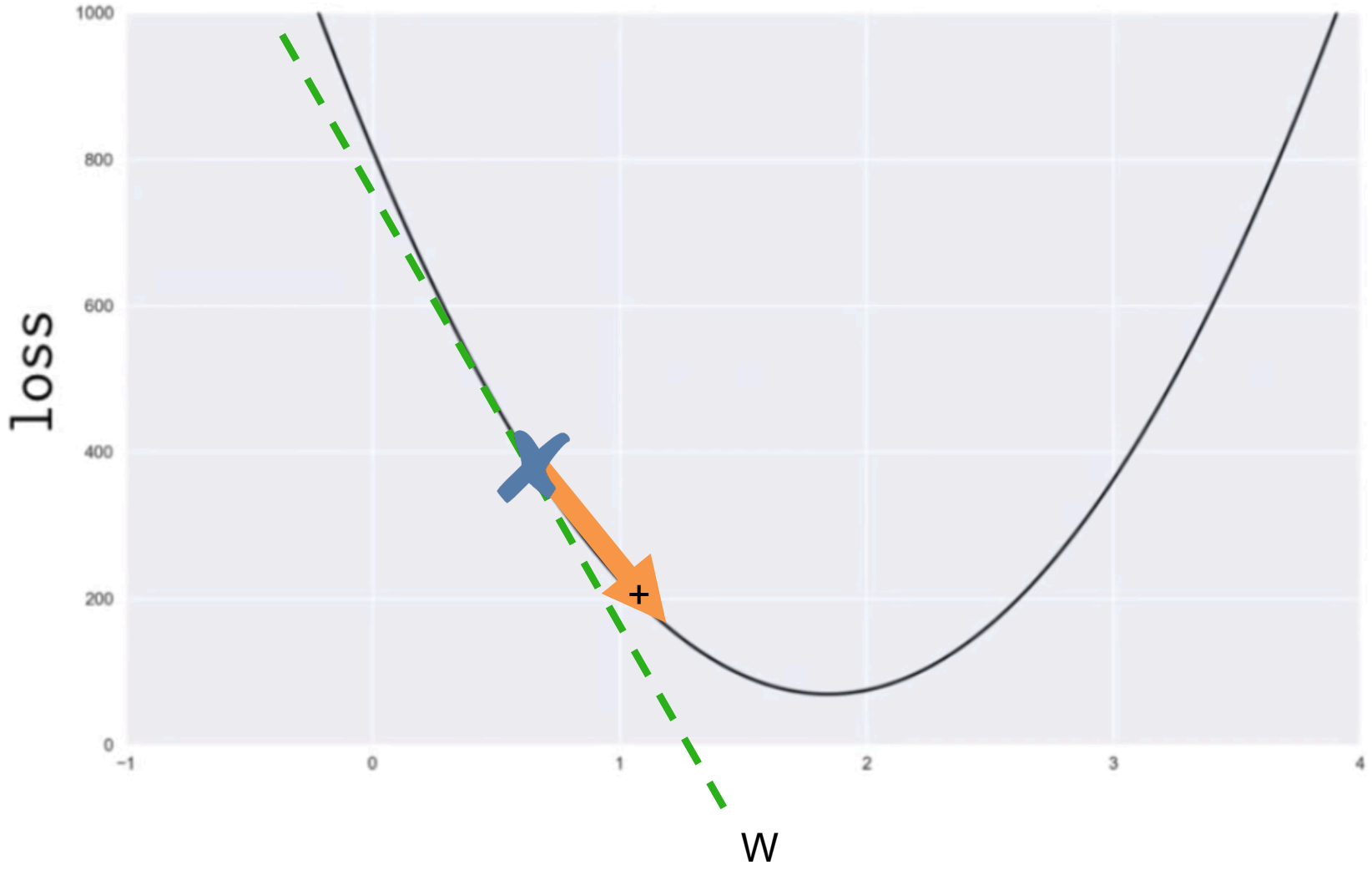
GRADIENT DESCENT



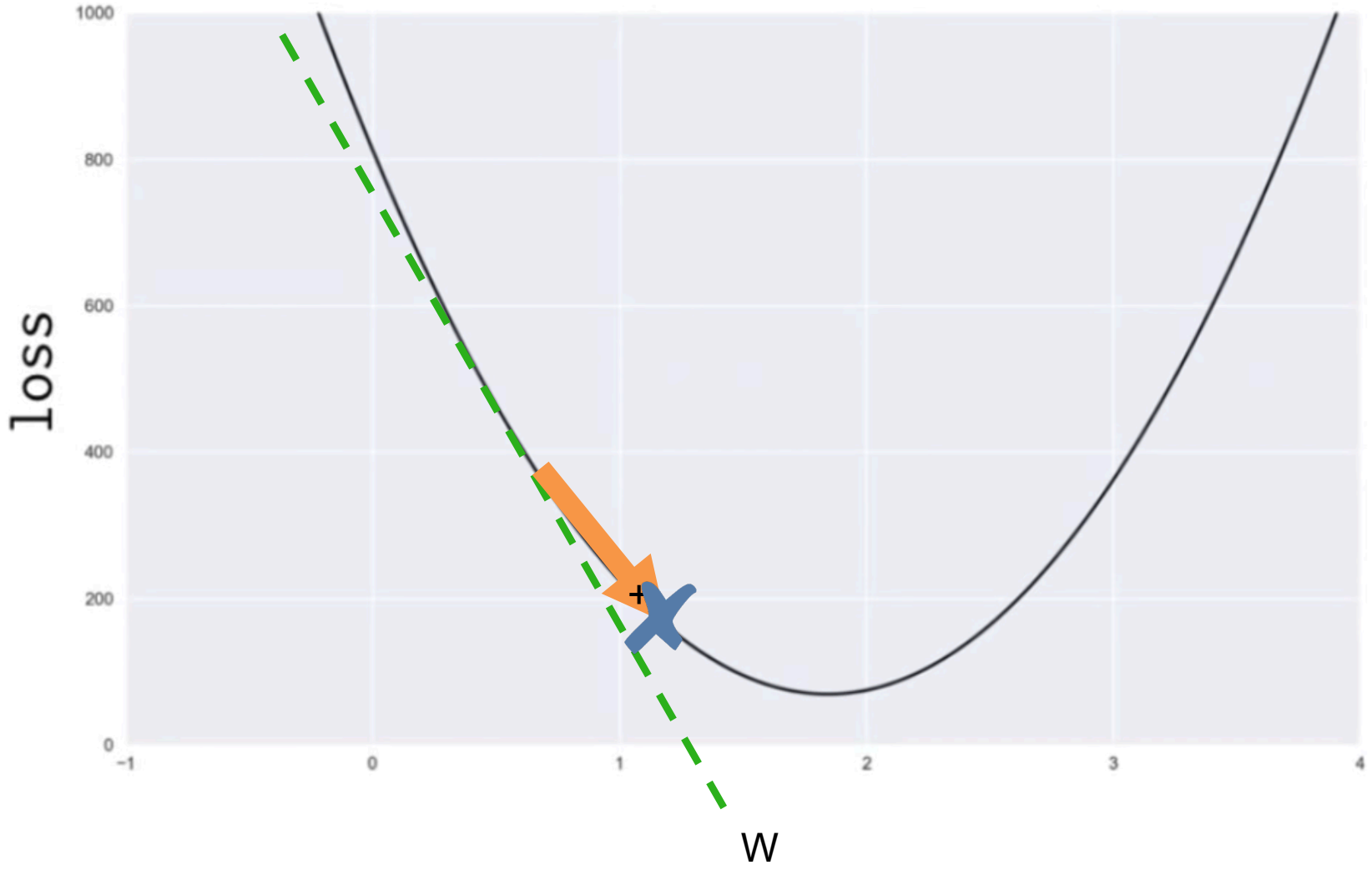
GRADIENT DESCENT



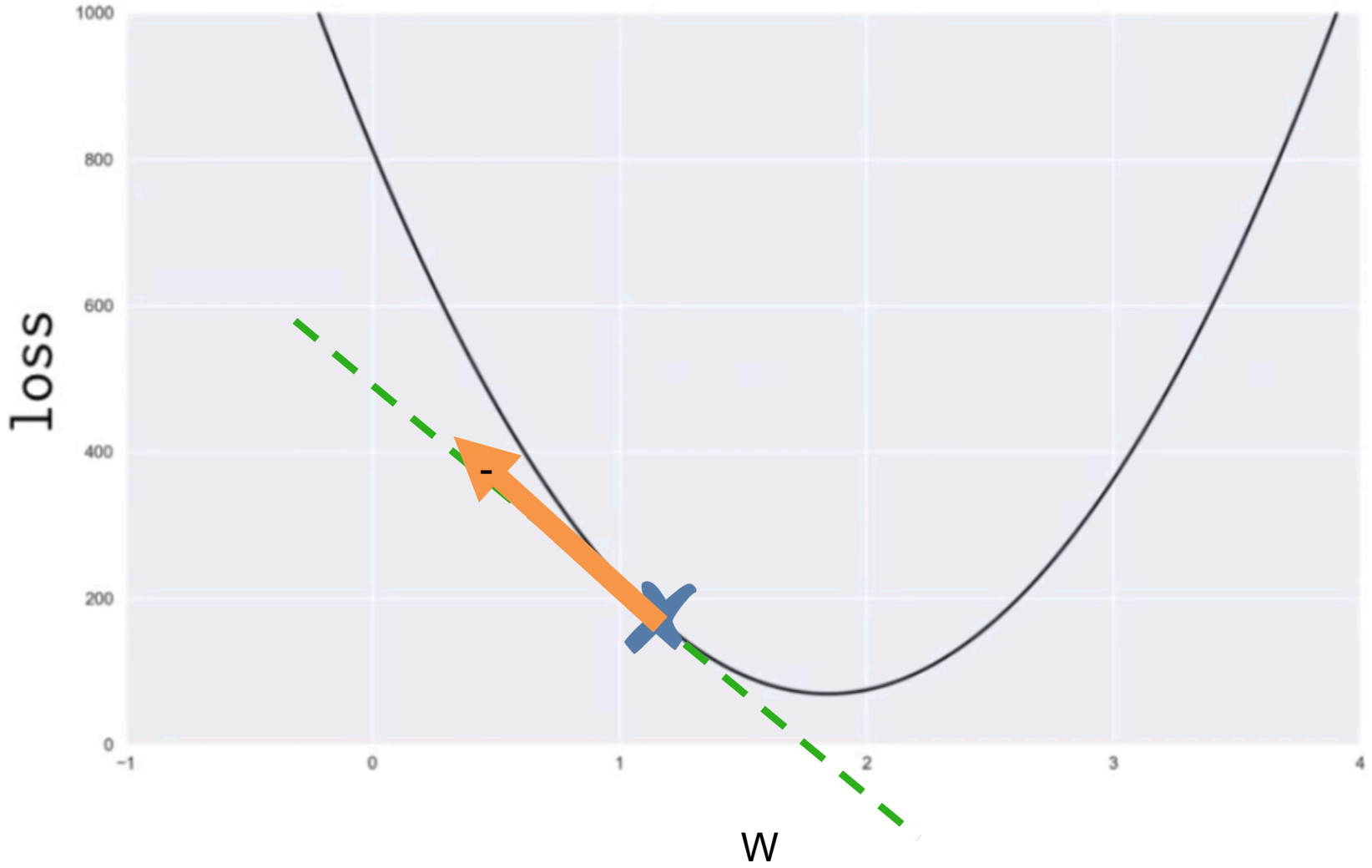
GRADIENT DESCENT



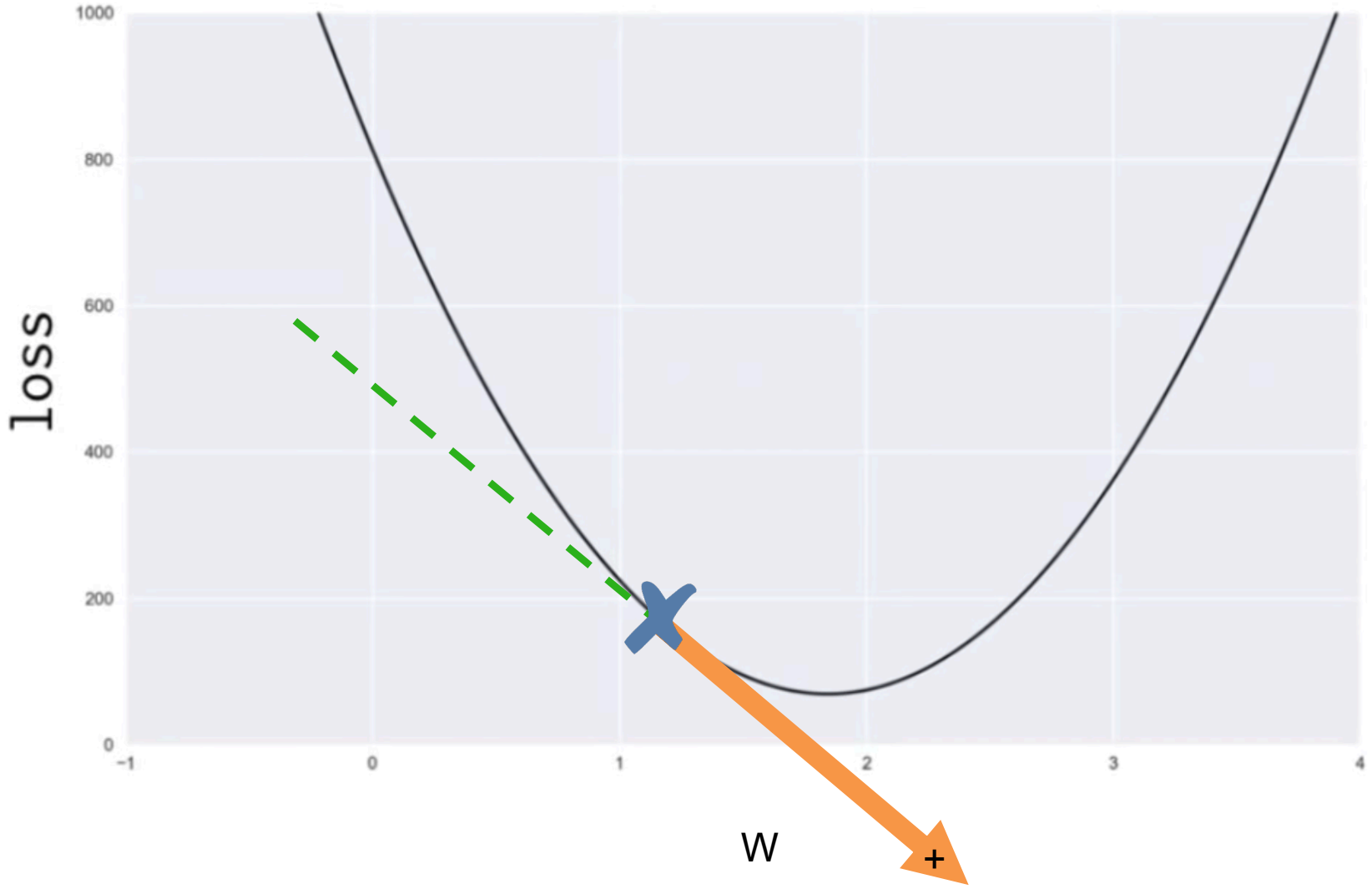
GRADIENT DESCENT



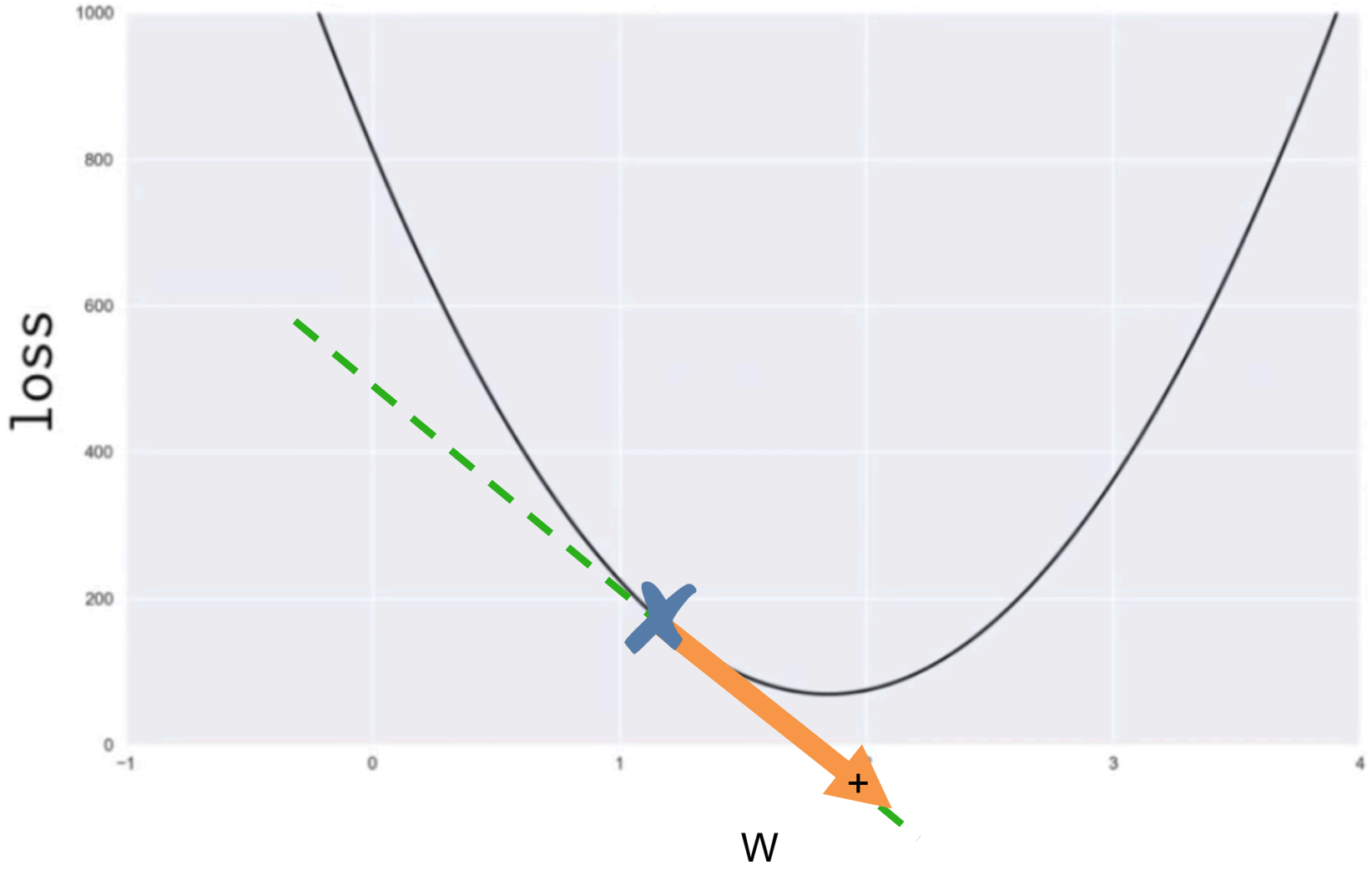
GRADIENT DESCENT



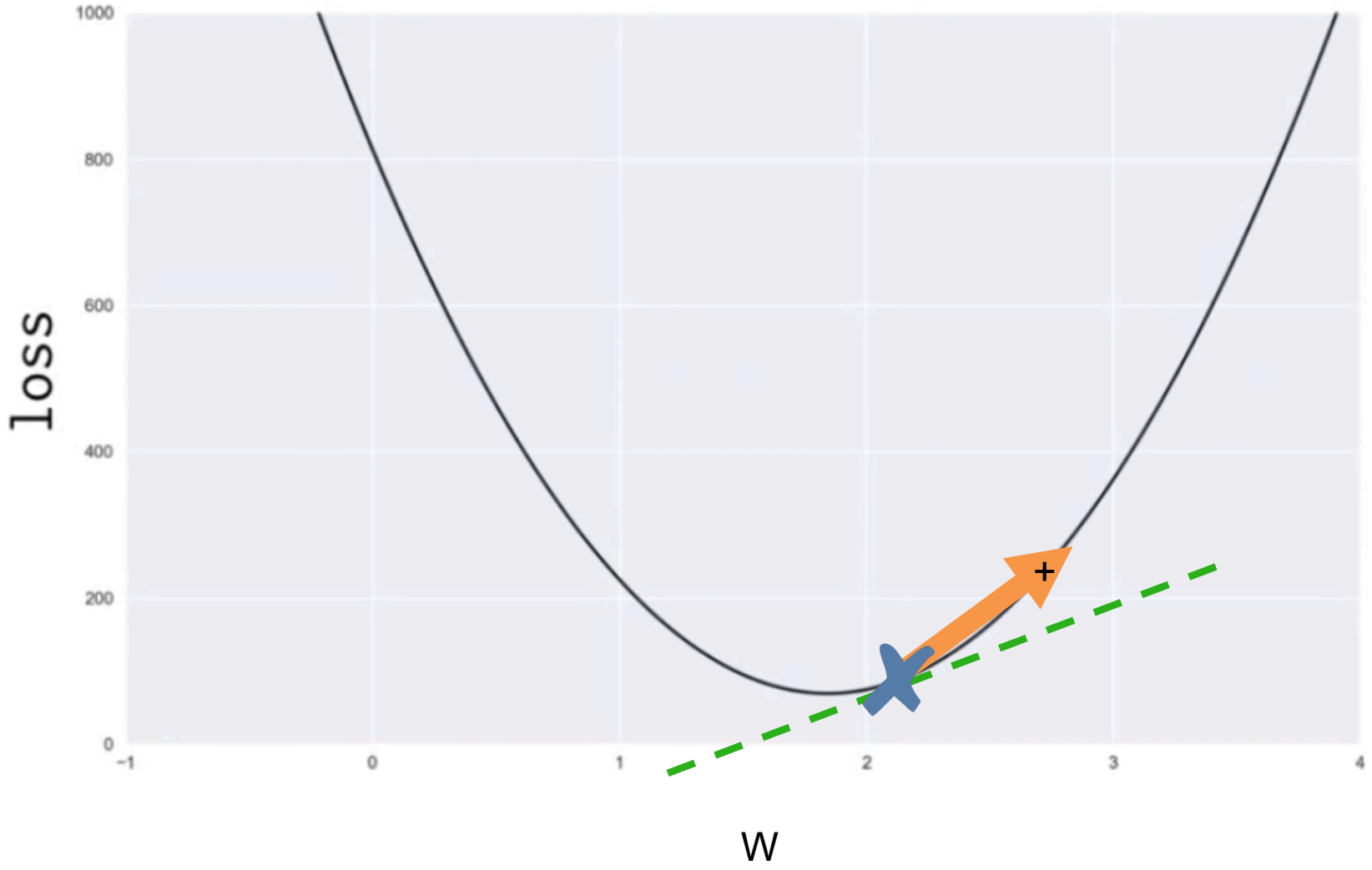
GRADIENT DESCENT



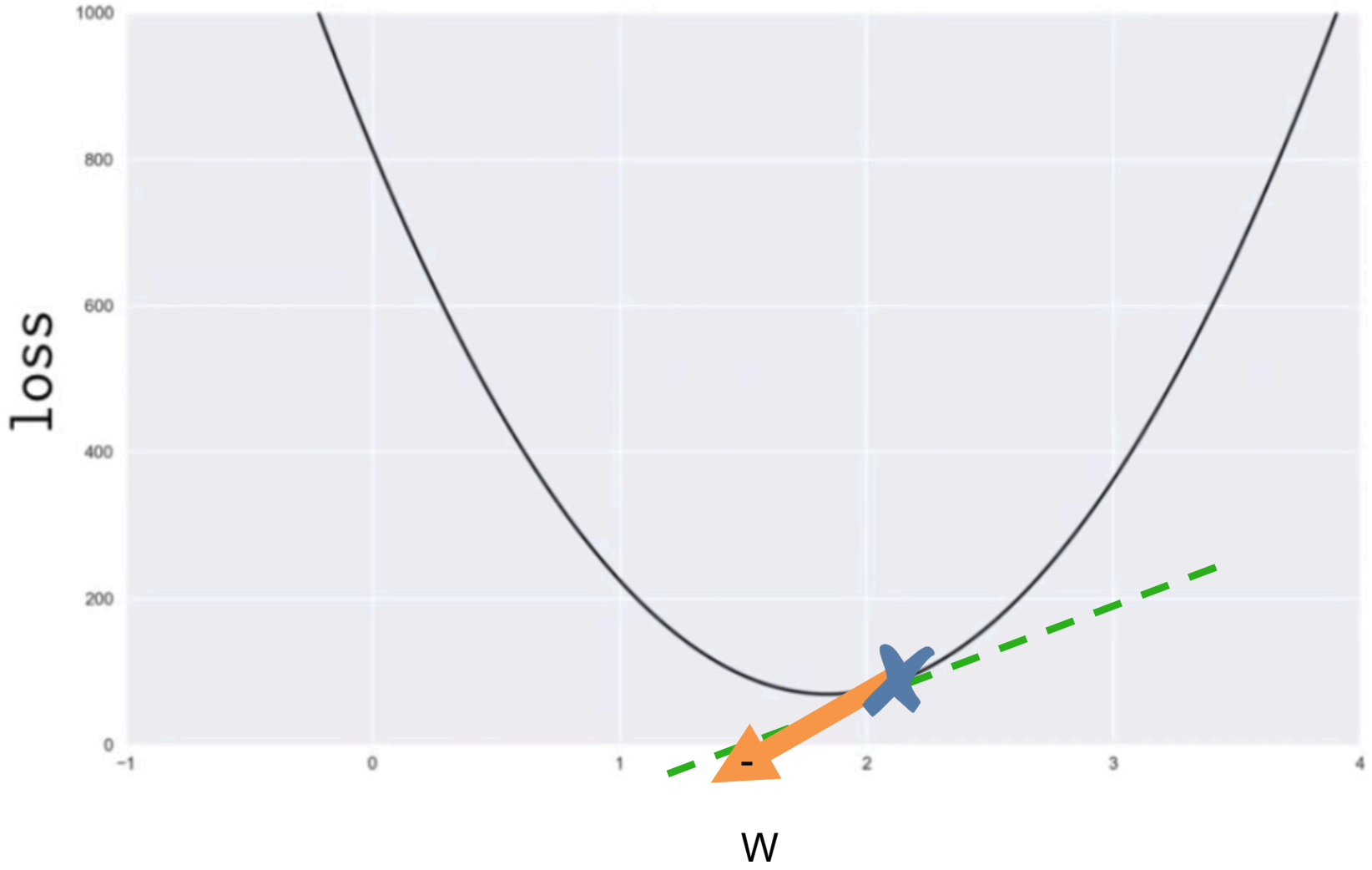
GRADIENT DESCENT



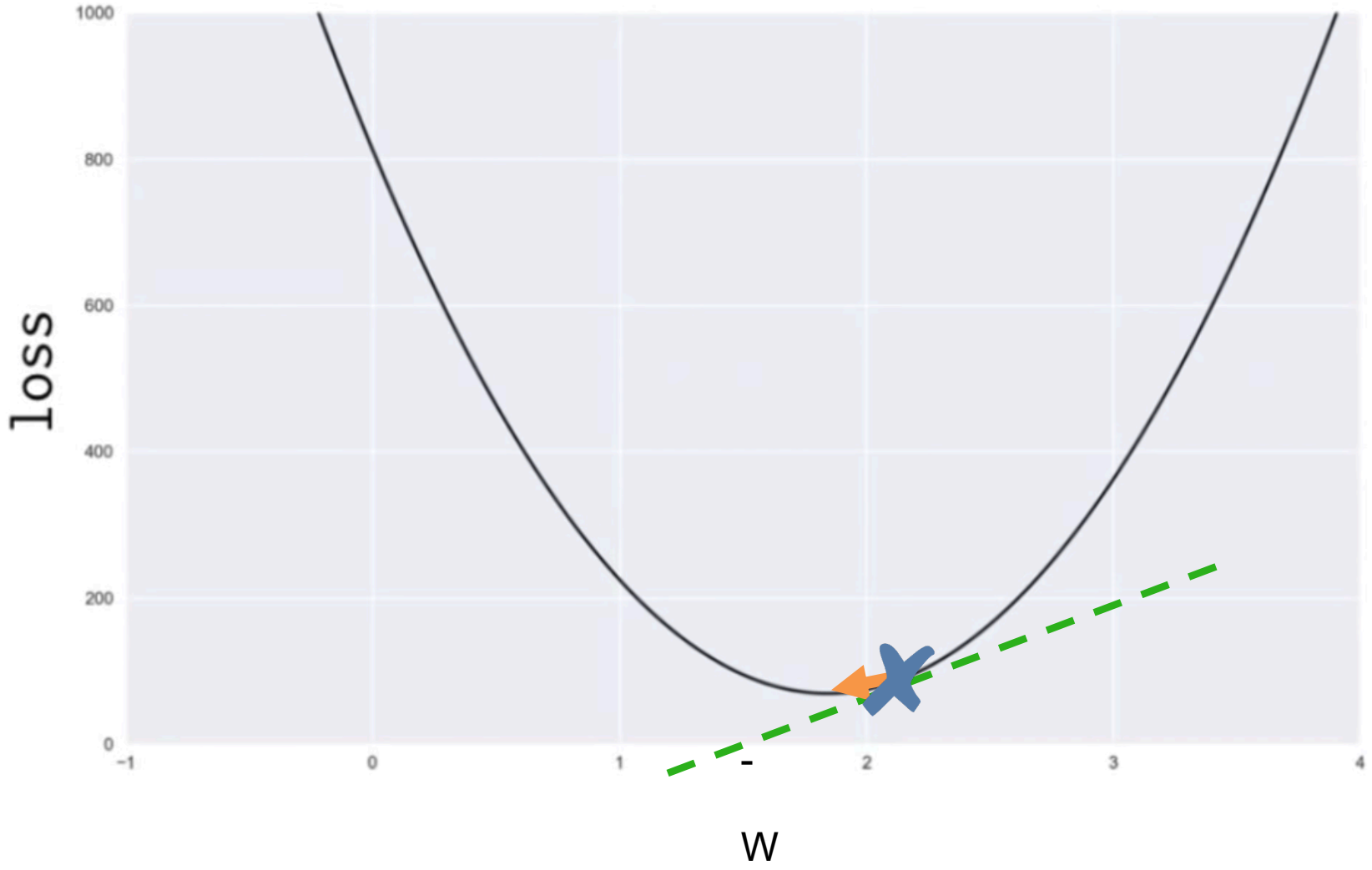
GRADIENT DESCENT



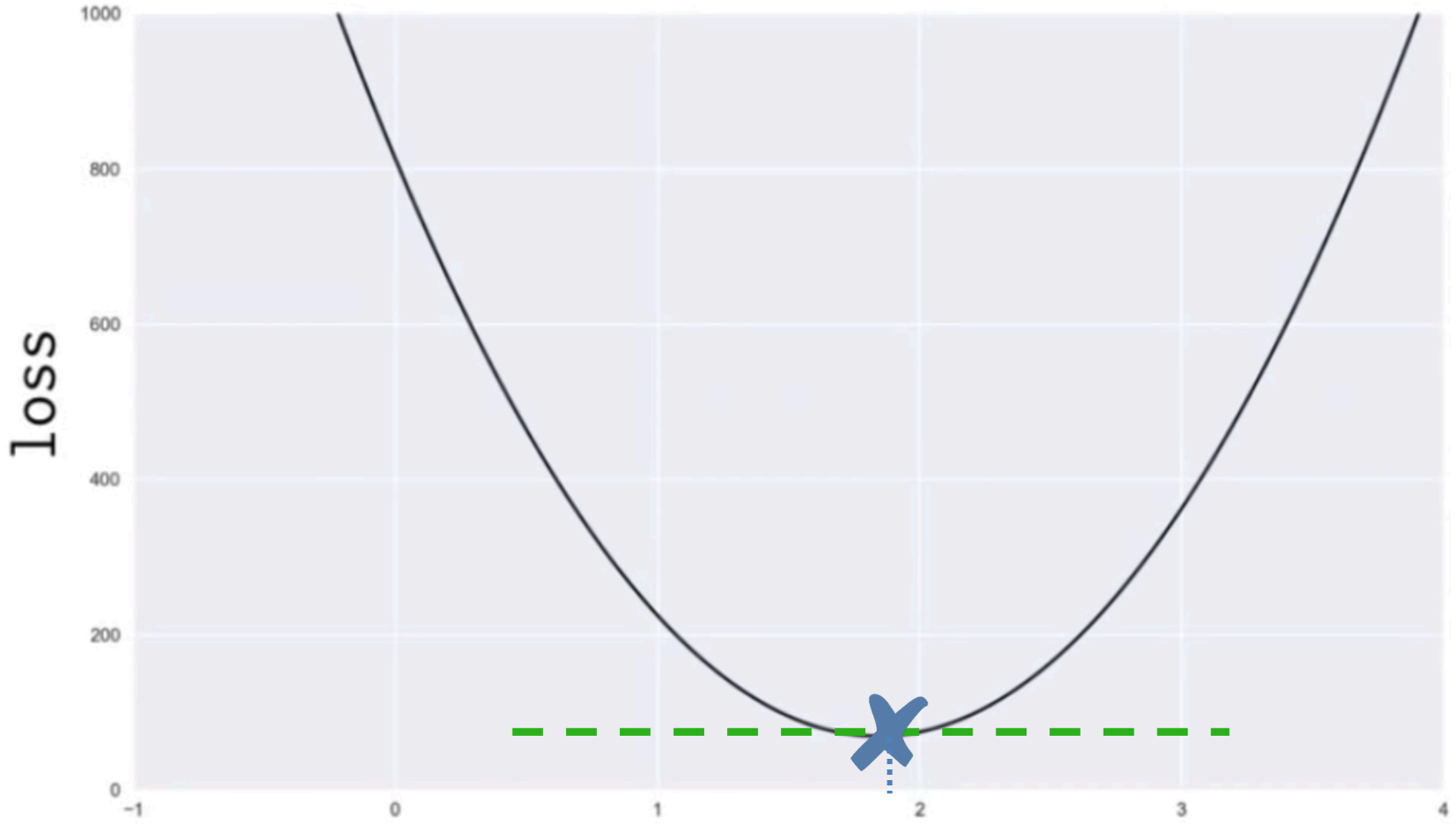
GRADIENT DESCENT



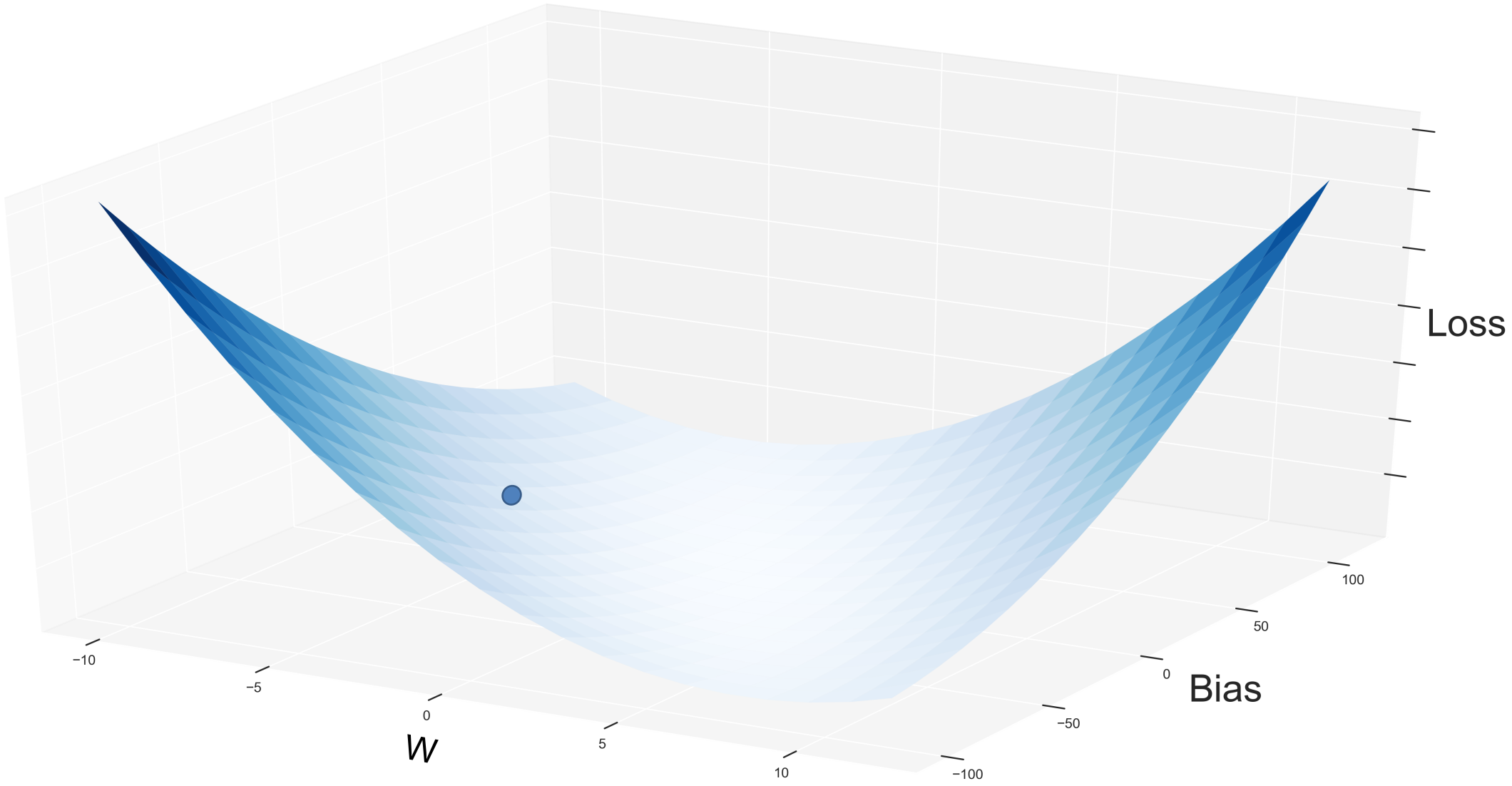
GRADIENT DESCENT



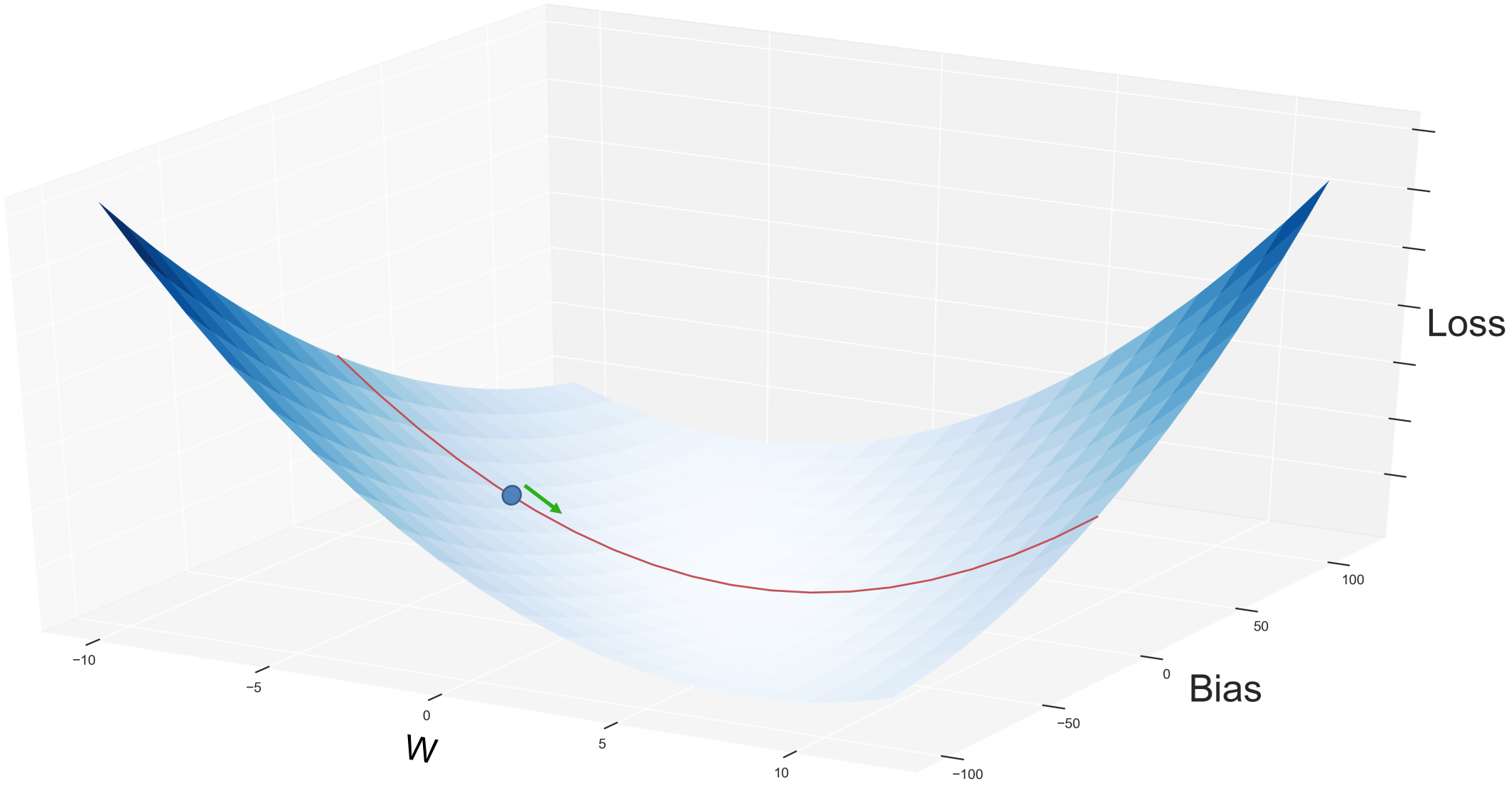
GRADIENT DESCENT



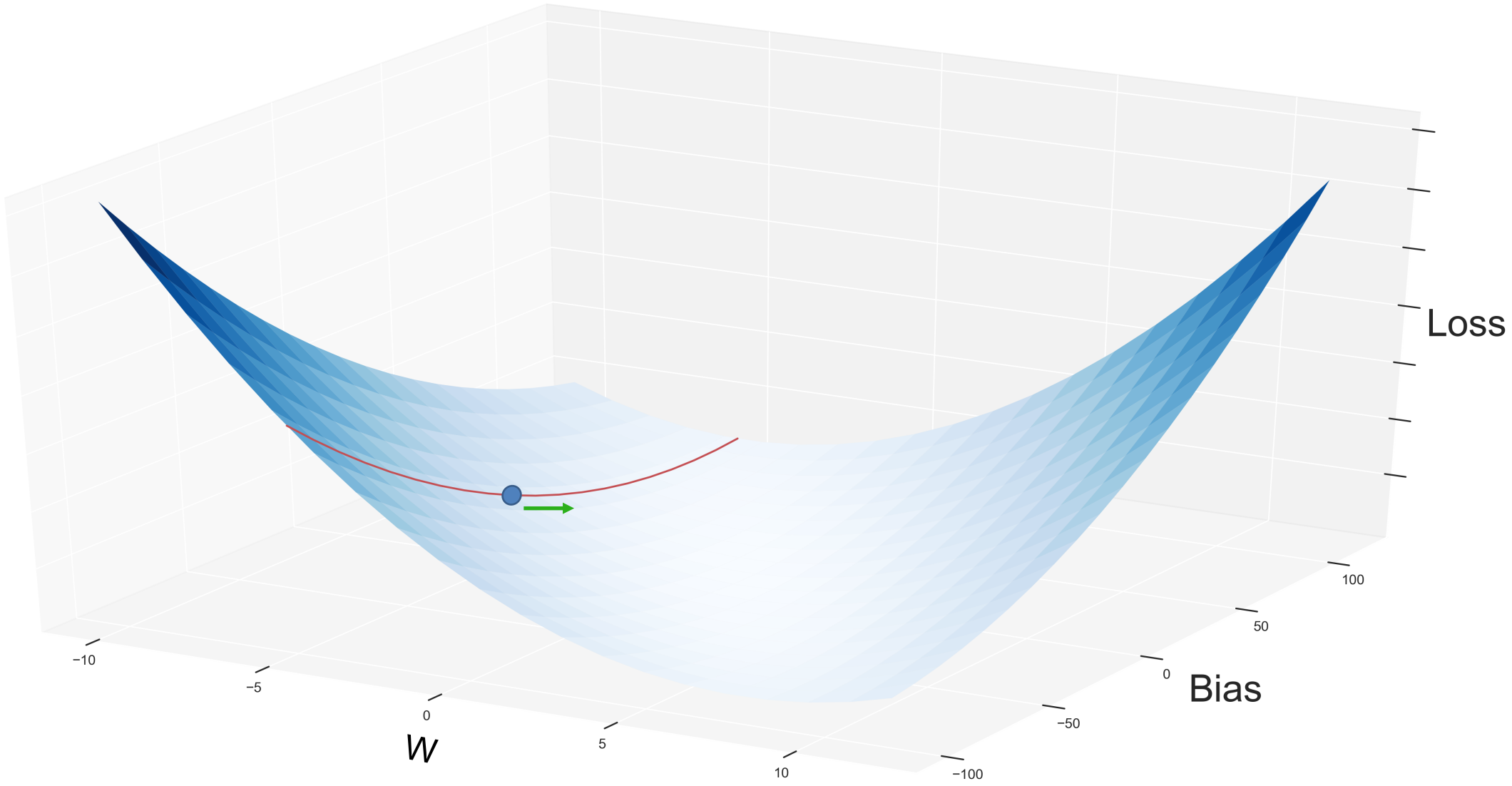
GRADIENT DESCENT



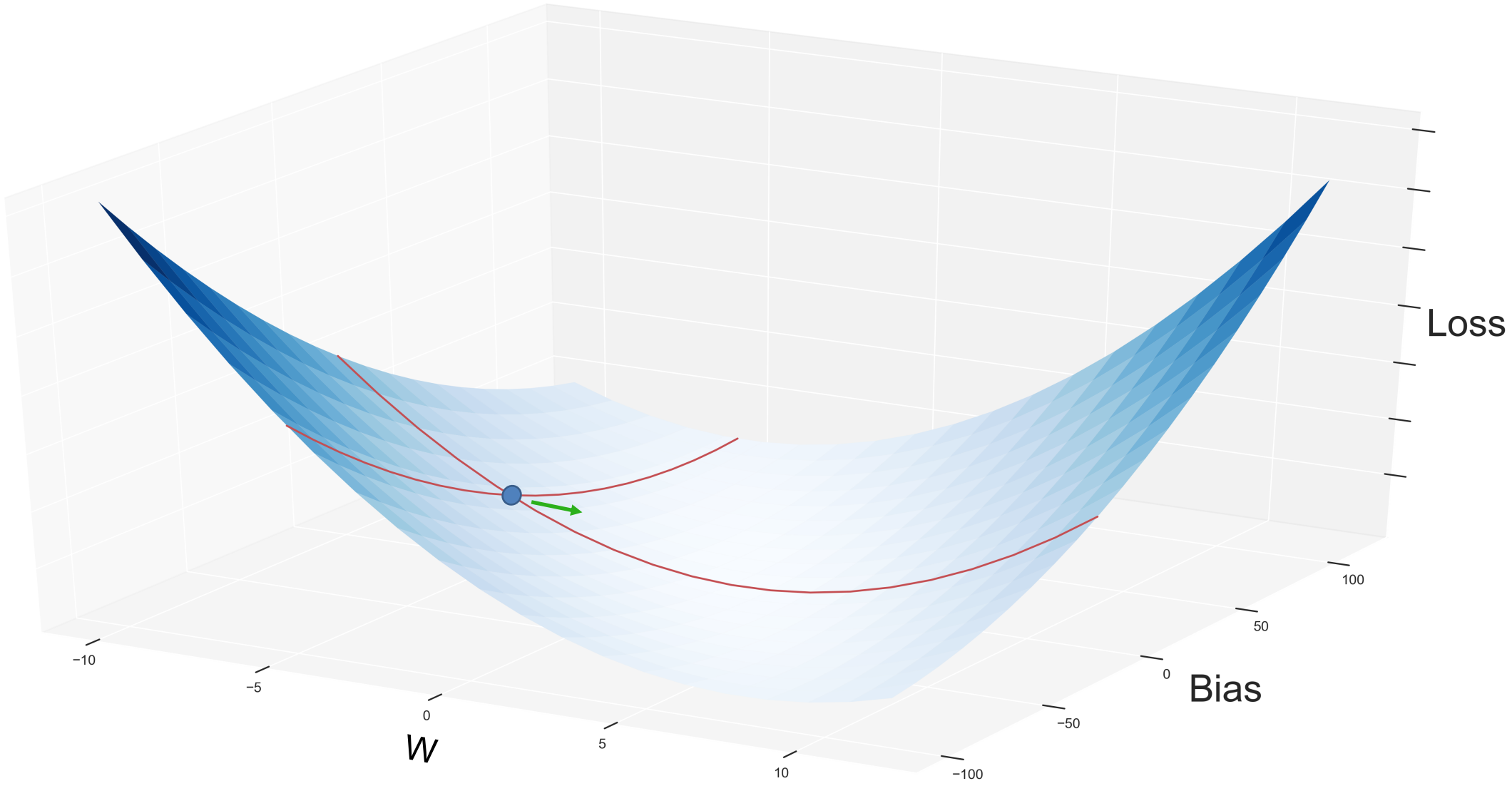
GRADIENT DESCENT



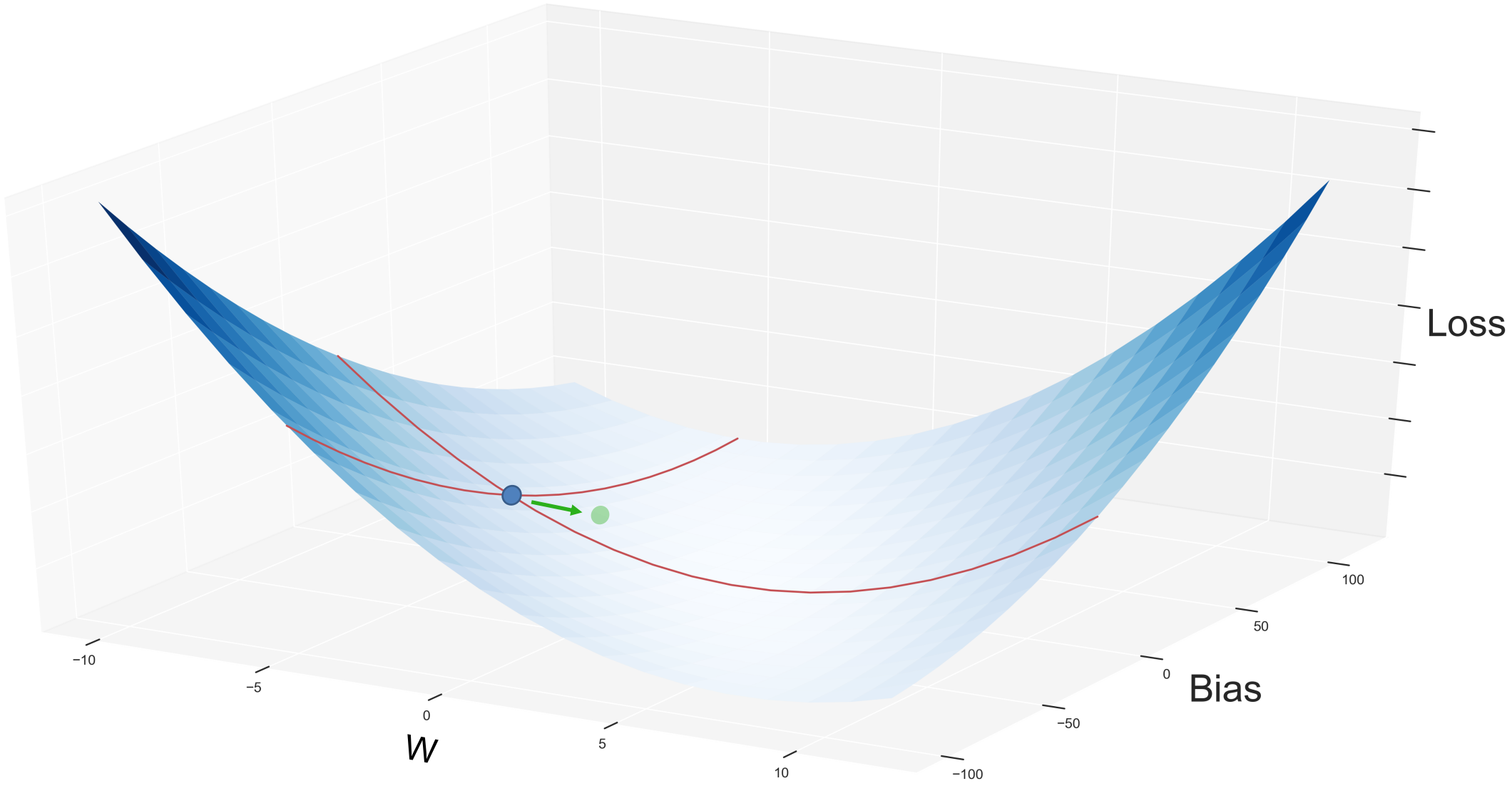
GRADIENT DESCENT



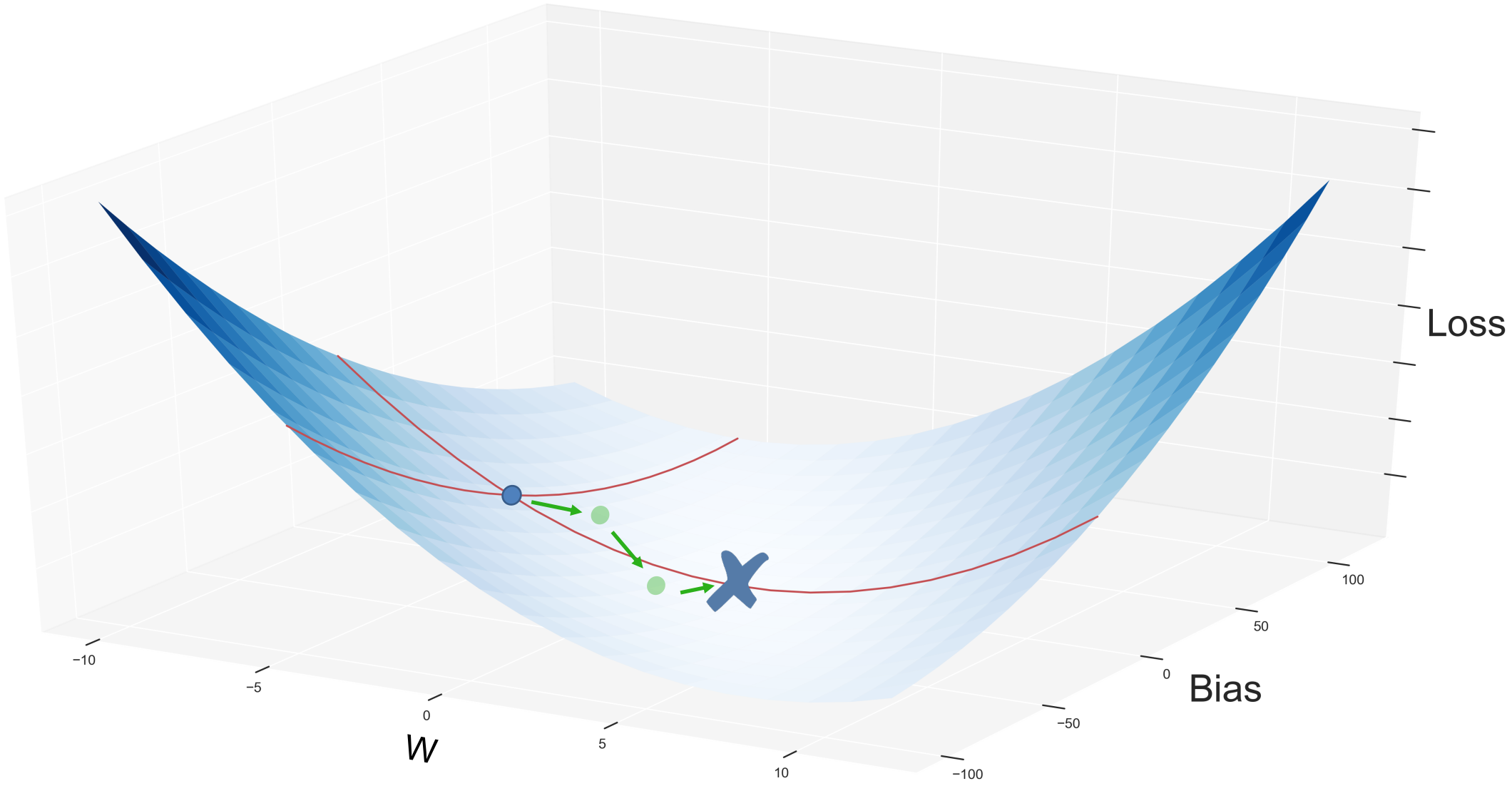
GRADIENT DESCENT



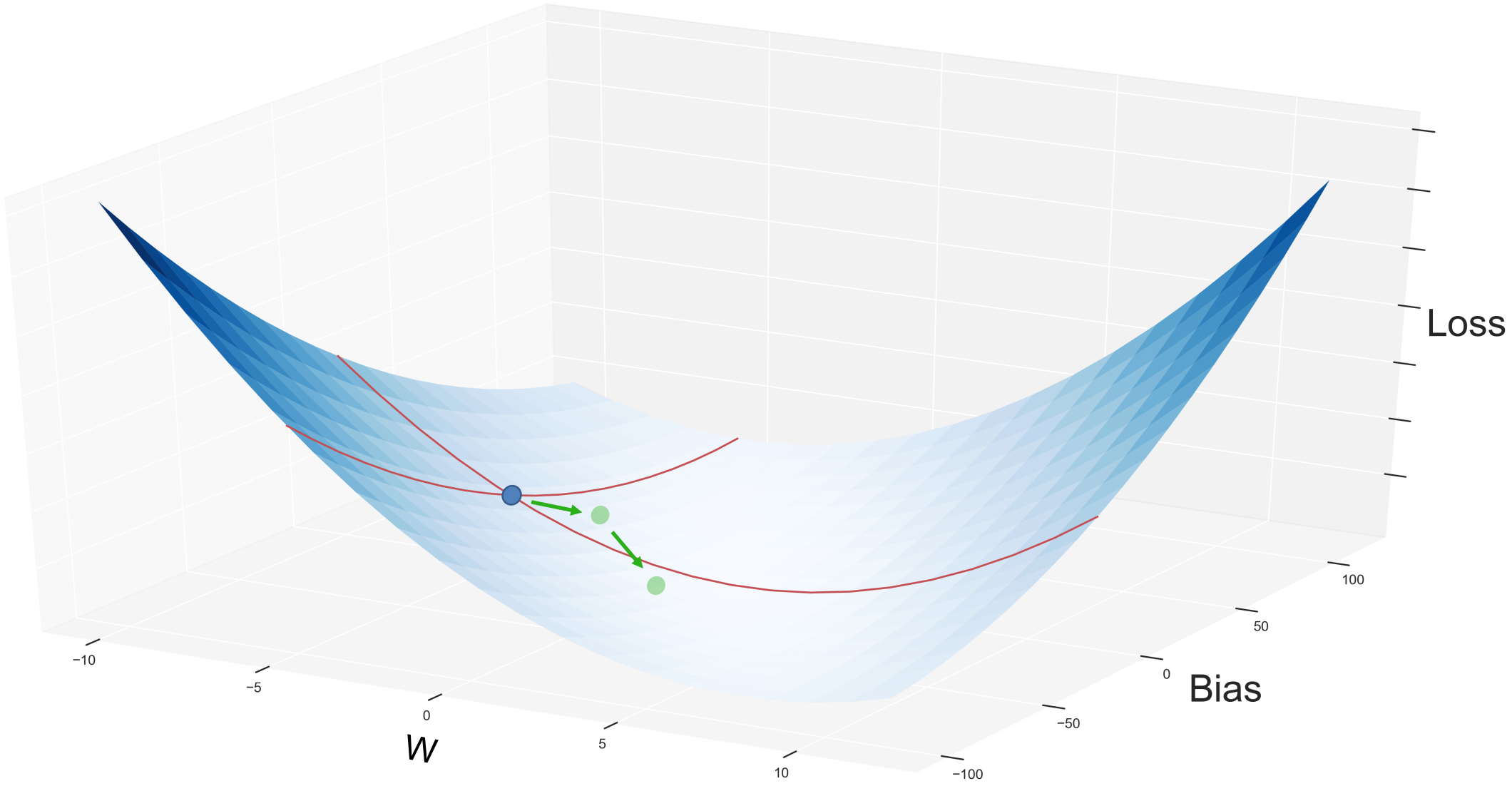
GRADIENT DESCENT



GRADIENT DESCENT



GRADIENT DESCENT



GRADIENT DESCENT

$$L = \frac{1}{m}\sum_{i=1}^{m}((wx_i + b) - y_i)^2$$

squared

error

mean

$$\frac{\partial L}{\partial w} = \frac{2}{m}\sum_{i=1}^{m} x_i((wx_i + b) - y_i)$$

$$\frac{\partial L}{\partial b} = \frac{2}{m}\sum_{i=1}^{m}((wx_i + b) - y_i)$$

GRADIENT DESCENT

```
def loss(X,Y,w,b):
    error=prediction(X,w,b) - Y
    squared_error = error**2
    return np.average(squared_error)
```

$$\frac{\partial L}{\partial w} = \frac{2}{m}\sum_{i=1}^{m} x_i((wx_i + b) - y_i)$$

$$\frac{\partial L}{\partial b} = \frac{2}{m}\sum_{i=1}^{m} ((wx_i + b) - y_i)$$

GRADIENT DESCENT

```
def loss(X,Y,w,b):
    error=prediction(X,w,b) - Y
    squared_error = error**2
    return np.average(squared_error)
```

```
def gradient(X, Y, w, b):
    w_gradient = 2 * np.average(X * (prediction(X, w, b) - Y))
    b_gradient = 2 * np.average(prediction(X, w, b) - Y)
    return (w_gradient, b_gradient)
```

Come riscrivo la mia funzione di training?

```python
def training(X,Y,iteration,learning_rate):
    w=0
    b=0
    for i in range(iteration):
        current_loss = loss(X,Y,w,b)
        print(f"iterazione {i+1} - w0: {w:2.3f} b0: {b:2.3f} - loss:{current_loss}")
        if(loss(X,Y,w+learning_rate,b)<current_loss):
            w+=learning_rate
        elif(loss(X,Y,w-learning_rate,b)<current_loss):
            w-=learning_rate
        elif(loss(X,Y,w,b+learning_rate)<current_loss):
            b+=learning_rate
        elif(loss(X,Y,w,b-learning_rate)<current_loss):
            b-=learning_rate
        elif(loss(X,Y,w+learning_rate,b-learning_rate)<current_loss):
            w+=learning_rate
            b-=learning_rate
        elif(loss(X,Y,w-learning_rate,b-learning_rate)<current_loss):
            w-=learning_rate
            b-=learning_rate
        elif(loss(X,Y,w+learning_rate,b+learning_rate)<current_loss):
            w+=learning_rate
            b+=learning_rate
        elif(loss(X,Y,w-learning_rate,b+learning_rate)<current_loss):
            w-=learning_rate
            b+=learning_rate
        else:
            return w,b

    raise Exception(f"Non sono riuscito a convergere con {iteration} iterazioni")
```

Come riscrivo la mia funzione di training?

```
def training(X,Y,iteration,learning_rate):
    w=0,b=0
    for i in range(iteration):
        current_loss = loss(X,Y,w,b)
        w_gradient,b_gradient=gradient(X,Y,w,b)
        w-=w_gradient*learning_rate
        b-=b_gradient*learning_rate
    return w,b
```

UN LUNGO VIAGGIO VERSO LA «COMPUTER VISION»

Regressione lineare

Regressione multipla

PIU' VARIABILI IN INGRESSO?

X1 = #PRENOTAZIONI	Y = #CORNETTI
1	4
1	3
3	2
3	1
5	3
8	7
10	3
4	3
1	1
1	0
1	0
1	2
…	…

PIU' VARIABILI IN INGRESSO ?

X1 = #PRENOTAZIONI	Y = #CORNETTI
1	4
1	3
3	2
3	1
5	3
8	7
10	3
4	3
1	1
1	0
1	0
1	2
…	…

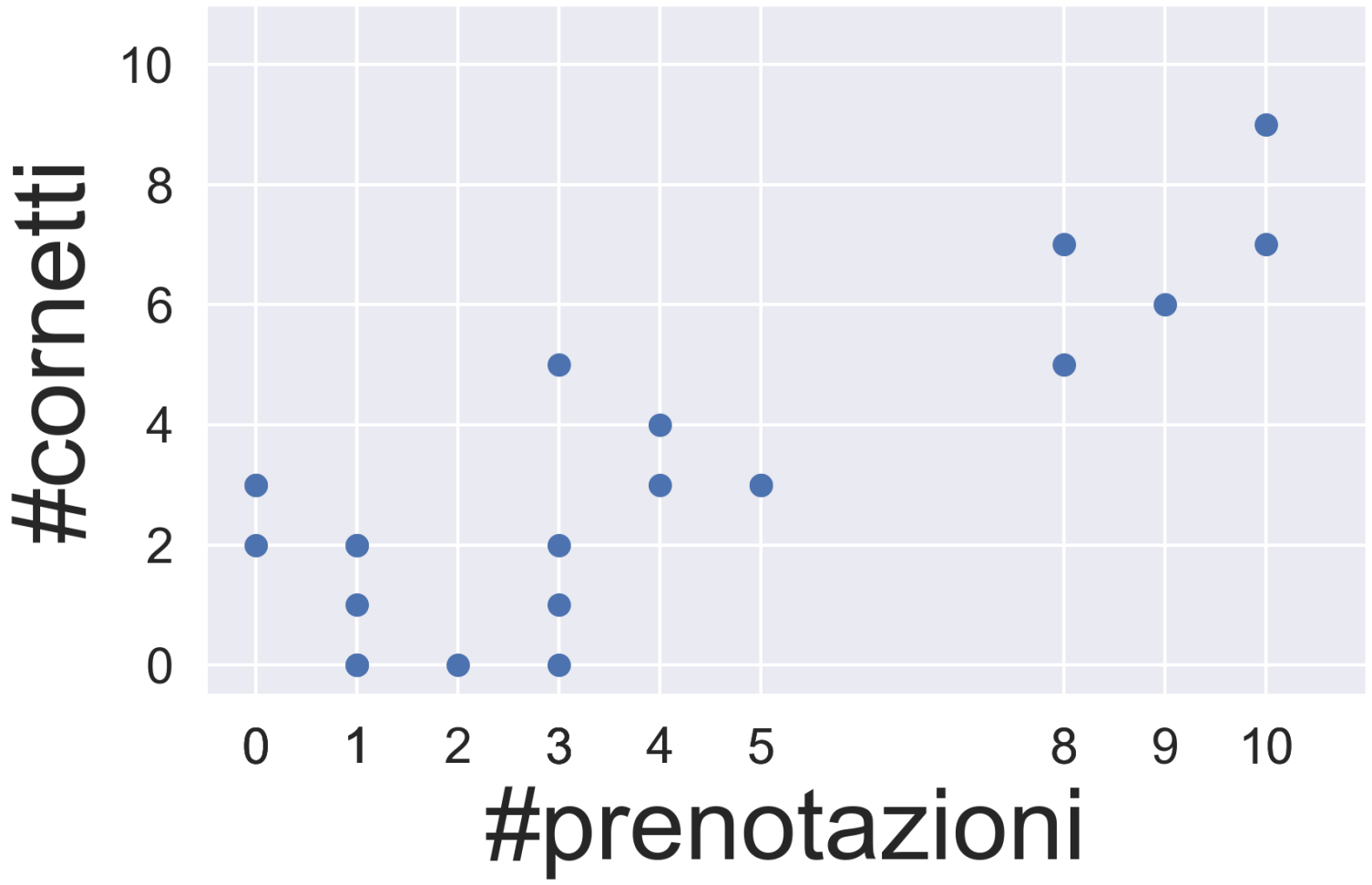
PIU' VARIABILI IN INGRESSO ?

X1 = #PRENOTAZIONI	X2= TEMPERATURA	Y = #CORNETTI
1	14	4
1	17	3
3	18	2
3	30	1
5	25	3
8	25	7
10	39	3
4	17	3
1	18	1
1	19	0
1	17	0
1	20	2
…	…	…

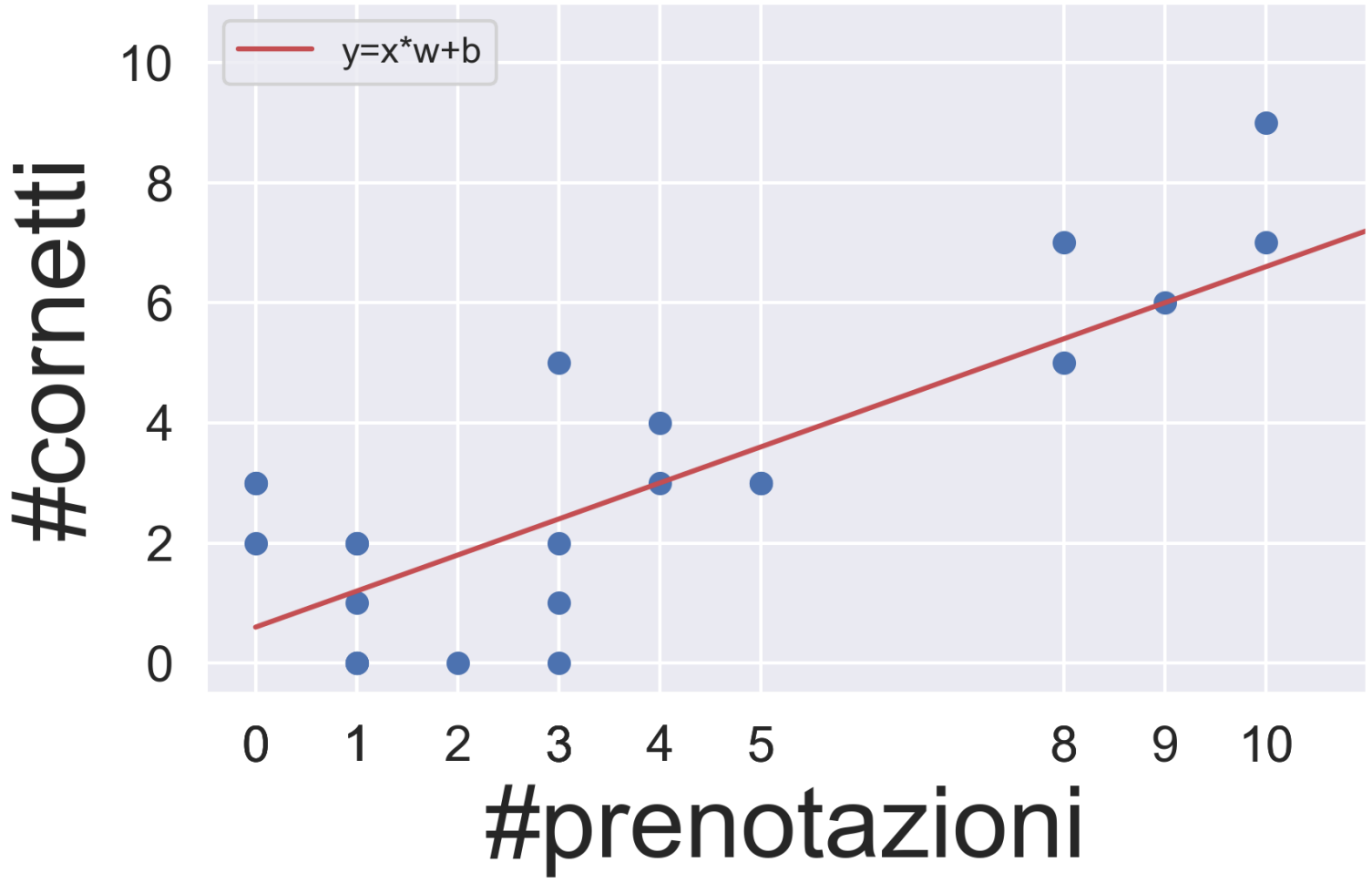
MATRICE

Come lo rappresento in un **grafico**?

UNA SOLA VARIABILE IN INGRESSO



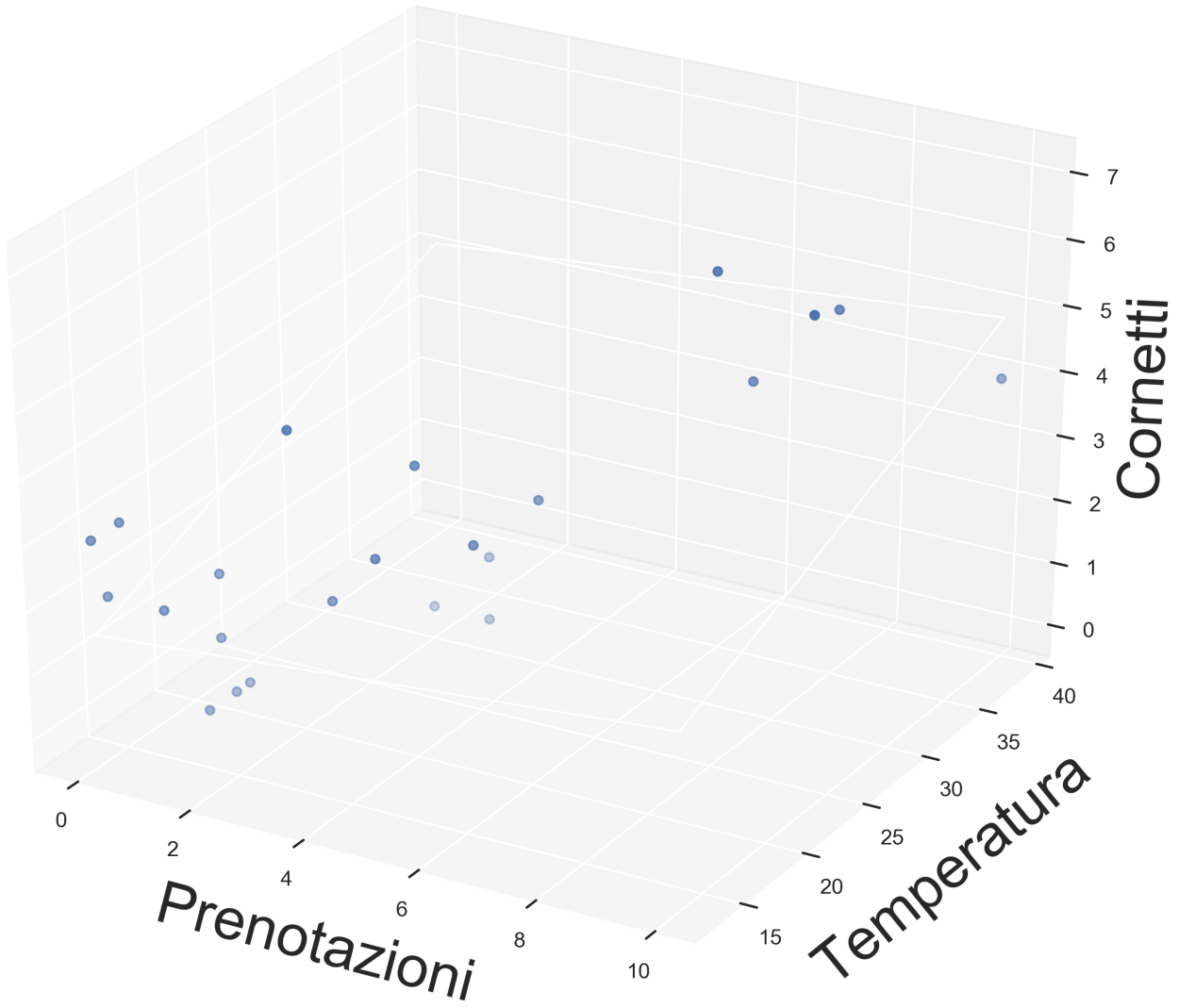
REGRESSIONE LINEARE



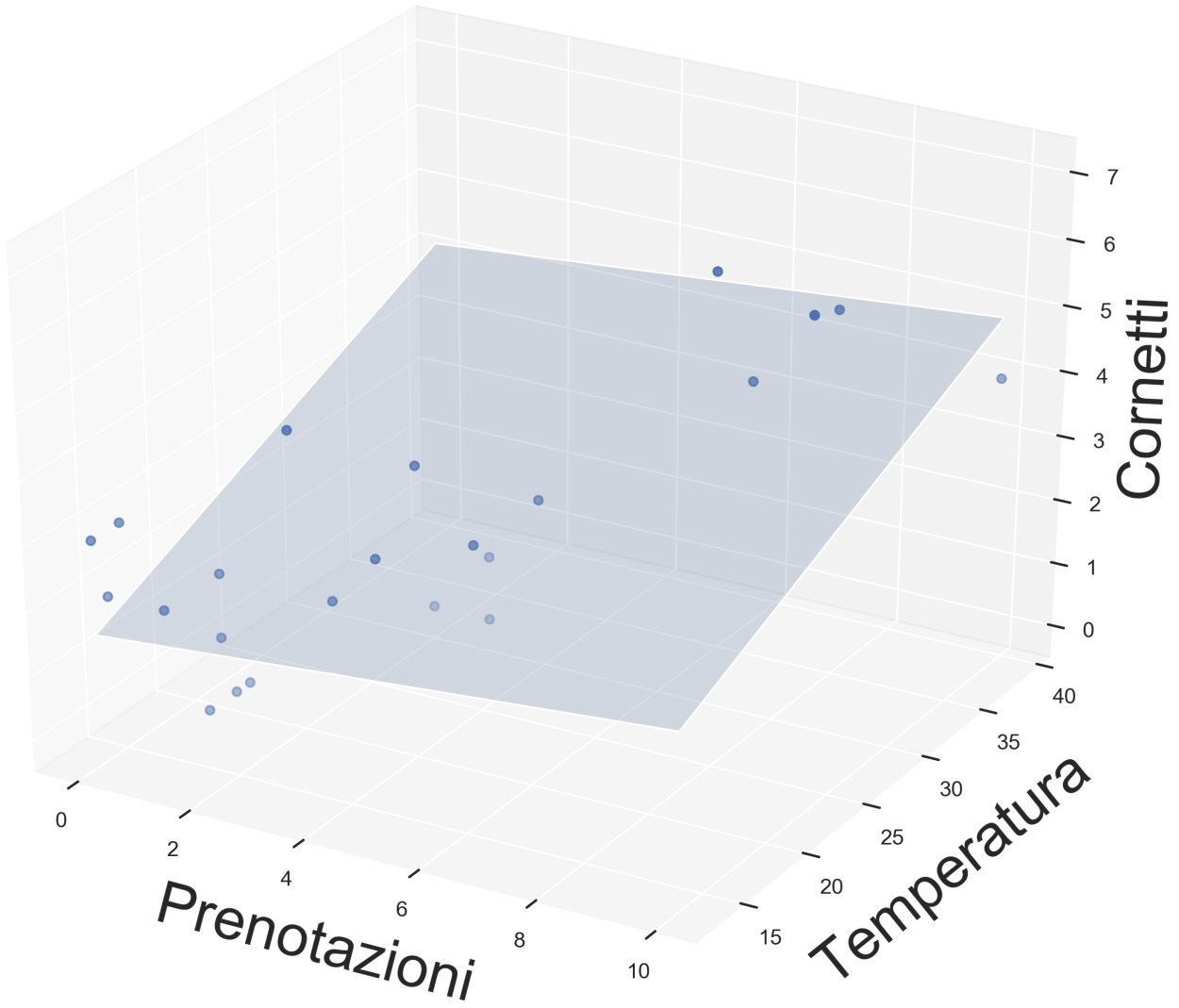
REGRESSIONE LINEARE

y=x***w** + **b**

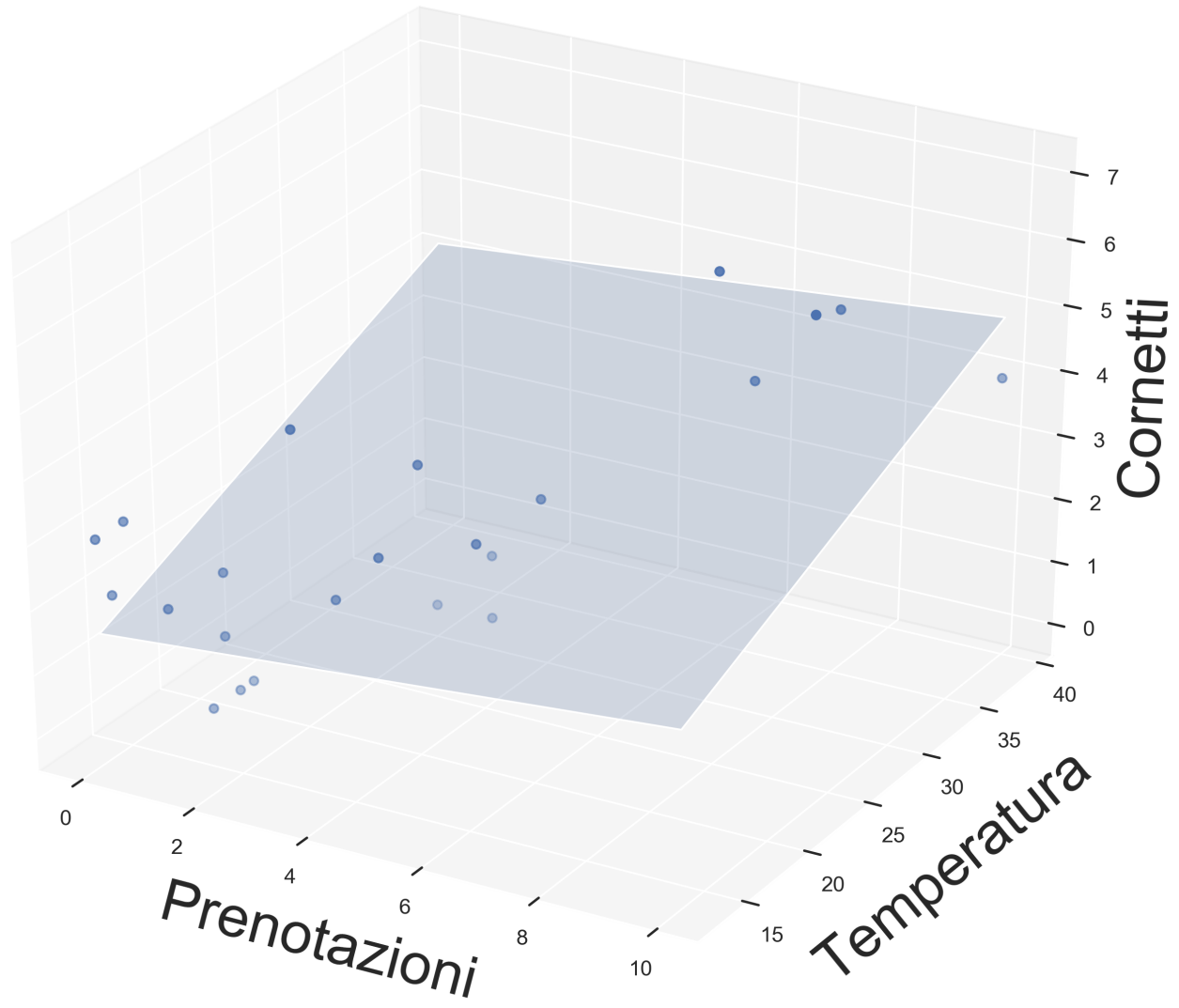
PIU' VARIABILI IN INGRESSO



PIU' VARIABILI IN INGRESSO



PIU' VARIABILI IN INGRESSO



regressione lineare

$\hat{y} = w1*\mathbf{x1} + b$

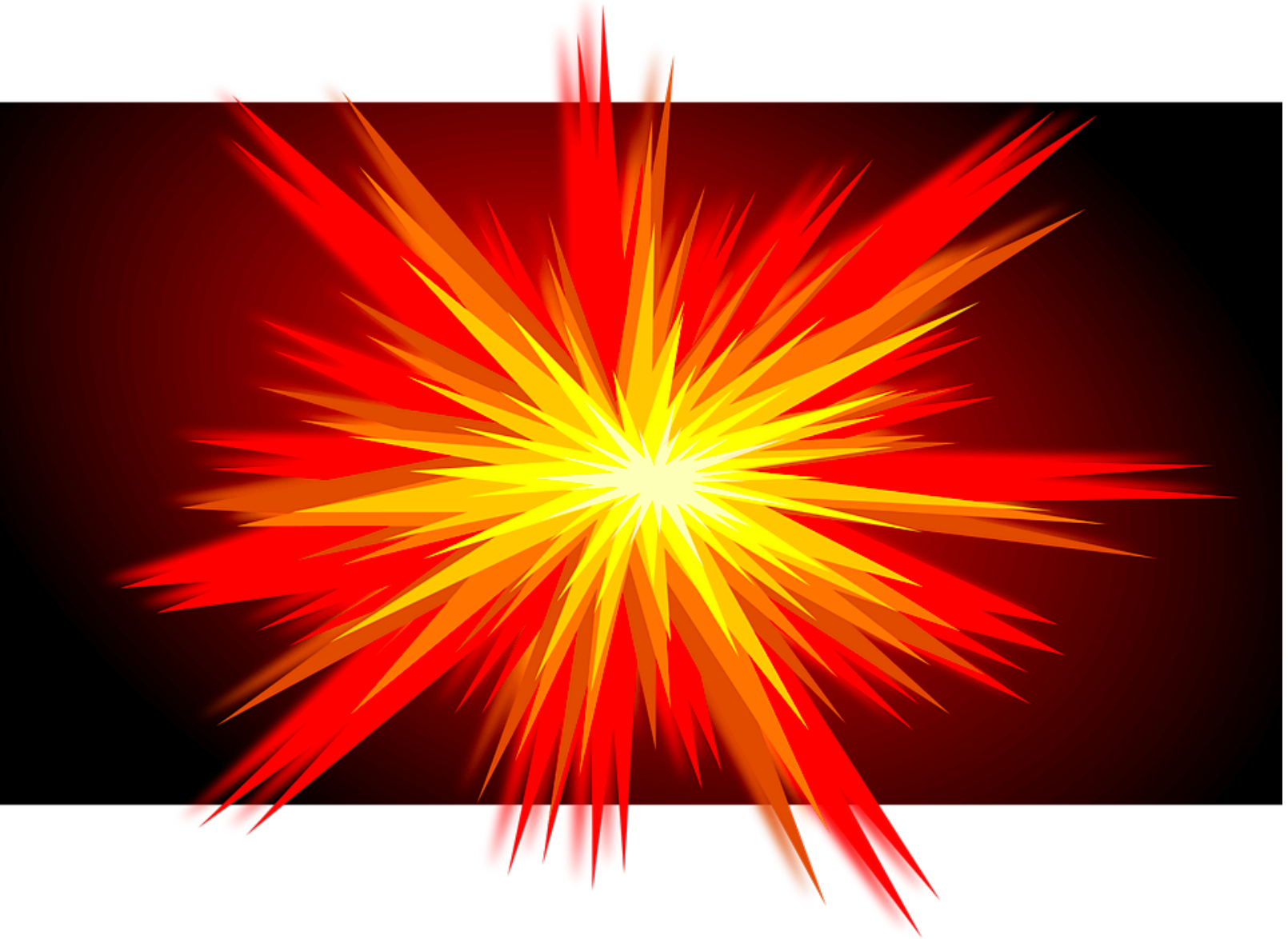
regressione lineare multipla:

$\hat{y} = w1*\mathbf{x1} + w2*\mathbf{x2} + b$

…e se avessi 3 variabili come lo disegnerei?

PIU' VARIABILI IN INGRESSO

Non riuscirei ad immaginarlo…



regressione lineare:

$\hat{y} = w1 * \mathbf{x1} + b$

regressione lineare multipla:

$\hat{y} = w1 * \mathbf{x1} + w2 * \mathbf{x2} + b$

…e se avessi 3 variabili come lo disegnerei?

PIU’ VARIABILI IN INGRESSO

PERO’…

PIU' VARIABILI IN INGRESSO

$$\hat{\mathbf{y}} = w1*\mathbf{x1} + w2*\mathbf{x2} + b$$

PIU’ VARIABILI IN INGRESSO

$$\hat{\mathbf{y}} = w1*\mathbf{x1} + w2*\mathbf{x2} + w3*\mathbf{x3} + b$$

PIU' VARIABILI IN INGRESSO

$$\hat{y} = w1*\mathbf{x1} + w2*\mathbf{x2} + w3*\mathbf{x3} + \ldots + b$$

REGRESSIONE LINEARE MULTIPLA

Quando con un CERTO NUMERO DI VARIABILI

x1, x2 , x3…

posso per spiegare una VARIABILE

$\hat{y}$

REGRESSIONE LINEARE MULTIPLA

$$\hat{\mathbf{y}} = w1*\mathbf{x1} + w2*\mathbf{x2} + w3*\mathbf{x3} + \ldots + b$$

TRANSFORMIAMO b

$$\hat{\mathbf{y}} = w1*\mathbf{x1} + w2*\mathbf{x2} + w3*\mathbf{x3} + \ldots + b$$

TRANSFORMIAMO b

$$\hat{\mathbf{y}} = b + w1*\mathbf{x1} + w2*\mathbf{x2} + w3*\mathbf{x3} + \ldots$$

TRANSFORMIAMO b

$$\hat{y} = b*\mathbf{1} + w1*\mathbf{x1} + w2*\mathbf{x2} + w3*\mathbf{x3} + \ldots$$

TRANSFORMIAMO b

$$\hat{y} = b*\mathbf{x0} + w1*\mathbf{x1} + w2*\mathbf{x2} + w3*\mathbf{x3} + \ldots$$

TRANSFORMIAMO b

$$\hat{y} = w0*\mathbf{x0} + w1*\mathbf{x1} + w2*\mathbf{x2} + w3*\mathbf{x3} + \ldots$$

$\hat{Y} = X * W$

X1 = #PRENOTAZIONI	X2= TEMPERATURA	Y = #CORNETTI
1	14	4
1	17	3
3	18	2
3	30	1
5	25	3
8	25	7
10	39	3
4	17	3
1	18	1
1	19	0
1	17	0
1	20	2
…	…	…

TRANSFORMIAMO b

$$\hat{y} = w0*\mathbf{x0} + w1*\mathbf{x1} + w2*\mathbf{x2} + w3*\mathbf{x3} + \ldots$$

$$\hat{\mathbf{Y}} = \mathbf{X} * \mathbf{W}$$

vettore Matrice Vettore

$$\hat{Y} = X * W$$

$\hat{Y} = X * W$

X1 = #PRENOTAZIONI	X2= TEMPERATURA	Y = #CORNETTI
1	14	4
1	17	3
3	18	2
3	30	1
5	25	3
8	25	7
10	39	3
4	17	3
1	18	1
1	19	0
1	17	0
1	20	2
…	…	…

$\hat{Y} = X * W$

X0 = Colonna Bias	X1 = #PRENOTAZIONI	X2= TEMPERATURA
1	1	14
1	1	17
1	3	18
1	3	30
1	5	25
1	8	25
1	10	39
1	4	17
1	1	18
1	1	19
1	1	17
1	1	20
…	…	…

X
matrice

$\hat{Y} = X * W$

X0 = Colonna Bias	X1 = #PRENOTAZIONI	X2= TEMPERATURA
1	1	14
1	1	17
1	3	18
1	3	30
1	5	25
1	8	25
1	10	39
1	4	17
1	1	18
1	1	19
1	1	17
1	1	20
…	…	…

X
matrice

*

pesi
w0
w1
w2

W
vettore

=

prediction (cornetti)

$\hat{Y}$
vettore

$\hat{Y} = X * W$

X0= Colonna Bias	X1 = #PRENOTAZIONI	X2= TEMPERATURA
1	1	14
1	1	17
1	3	18
1	3	30
1	5	25
1	8	25
1	10	39
1	4	17
1	1	18
1	1	19
1	1	17
1	1	20
…	…	…

X
matrice

*

pesi
w0
w1
w2

W
vettore

=

prediction (cornetti)
x0*w0 + x1*w1 + x2*w2

$\hat{Y}$
vettore

$\hat{Y} = X * W$

X0= Colonna Bias	X1 = #PRENOTAZIONI	X2= TEMPERATURA
1	1	14
1	1	17
1	3	18
1	3	30
1	5	25
1	8	25
1	10	39
1	4	17
1	1	18
1	1	19
1	1	17
1	1	20
…	…	…

X
matrice

*

pesi
w0
w1
w2

W
vettore

=

prediction (cornetti)
x0*w0 + x1*w1 + x2*w2
x0*w0 + x1*w1 + x2*w2

$\hat{Y}$
vettore

$\hat{Y} = X * W$

X0 = Colonna Bias	X1 = #PRENOTAZIONI	X2= TEMPERATURA
1	1	14
1	1	17
1	3	18
1	3	30
1	5	25
1	8	25
1	10	39
1	4	17
1	1	18
1	1	19
1	1	17
1	1	20
…	…	…

X
matrice

*

pesi
w0
w1
w2

W
vettore

=

prediction (cornetti)
x0*w0 + x1*w1 + x2*w2
x0*w0 + x1*w1 + x2*w2
x0*w0 + x1*w1 + x2*w2
x0*w0 + x1*w1 + x2*w2
…
…
…
…
…
…
…
…
…

$\hat{Y}$
vettore

$\hat{Y} = X * W$

$$\underset{\text{matrice}}{X} * \underset{\text{vettore}}{W} = \underset{\text{vettore}}{\hat{Y}}$$

$\hat{Y} = X * W$

$$\underset{\text{vettore}}{\hat{\mathbf{Y}}} = \underset{\text{matrice}}{\mathbf{X}} * \underset{\text{vettore}}{\mathbf{W}}$$

recap

Regressione lineare

$$\hat{y} = \mathbf{x1}*w1 + b$$

```
def predict(X, w, b):
    return X * w + b
```

Regressione multipla

$$\hat{y} = \mathbf{x1}*w1 + \mathbf{x2}*w2 + \ldots + b$$

$$\hat{y} = \mathbf{x0}*w0 + \mathbf{x1}*w1 + \mathbf{x2}*w2 + \ldots$$

$$\hat{\mathbf{Y}} = \mathbf{X}*\mathbf{W}$$

vettore Matrice Vettore

```
def predict(X, w):
    return np.matmul(X, w)
```

CODE…

UN LUNGO VIAGGIO VERSO LA «COMPUTER VISION»

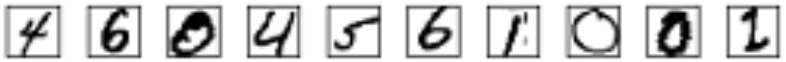
Regressione multipla

Regressione logistica

MACHINE LEARNING:
sistema di apprendimento supervisionato

TRAINING SET

TEST SET



[MNIST database]

scelgo un
MODELLO MATEMATICO

addestro

Modello matematico

astraggo il ***concetto***

predire

Modello predittivo

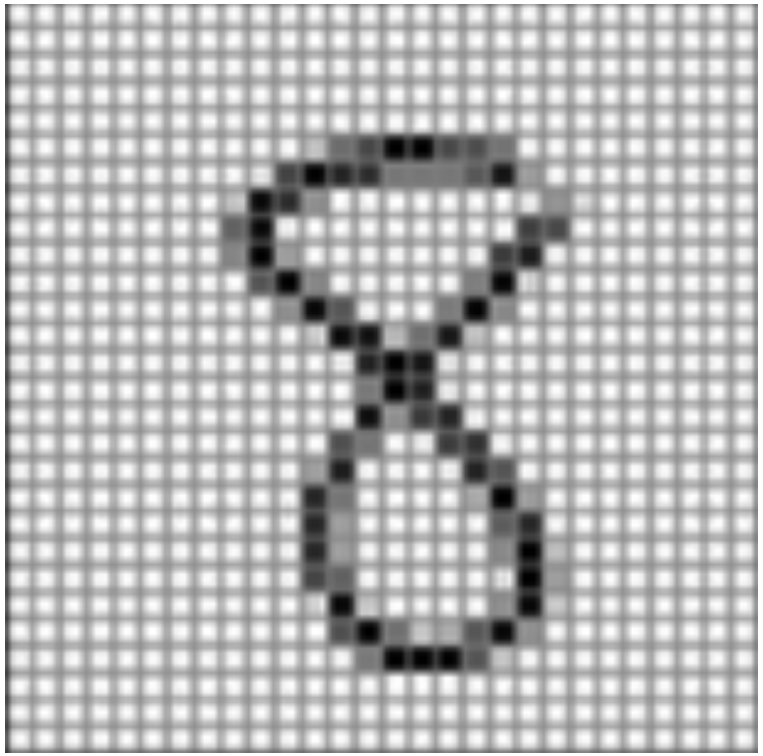
assegno

CLASSIFICATORE

label

- **zero**
- **uno**
- **due**
- **tre**
- **quattro**
- **cinque**
- **sei**
- **sette**
- **otto**
- **nove**

Label «otto»



28 x 28
784 pixel

70K digits

MNIST database

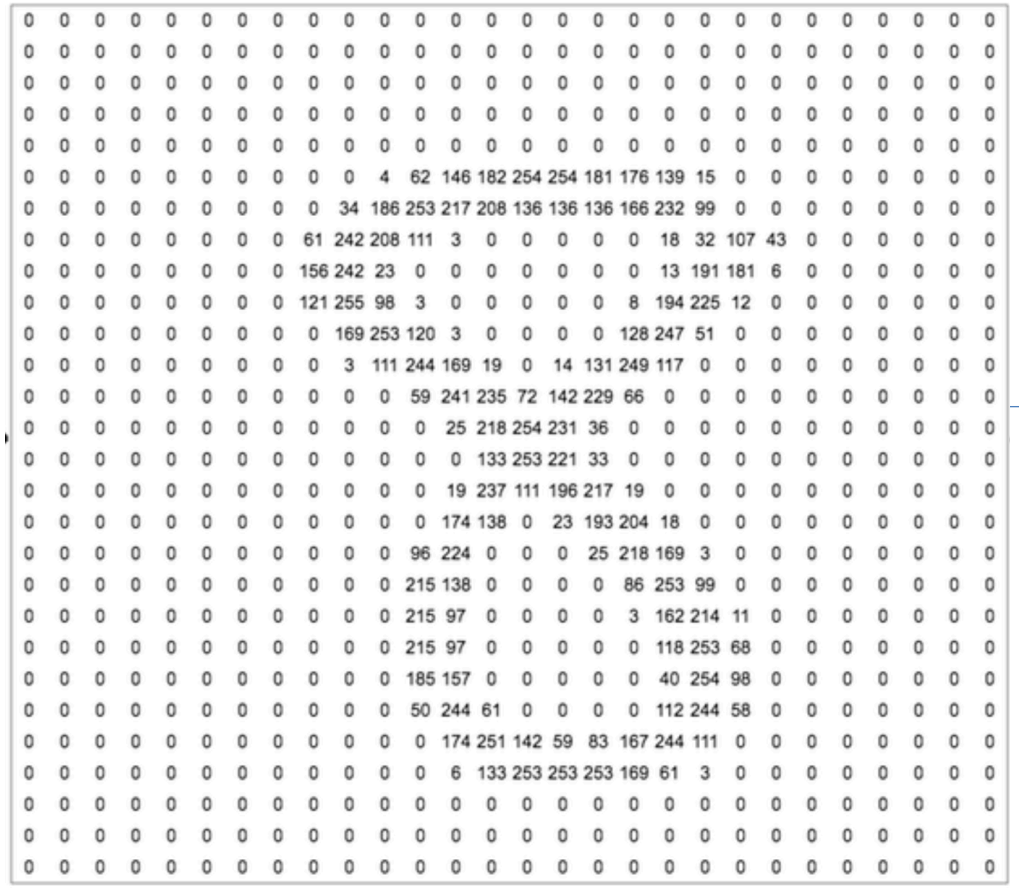
Label «otto»

28 x 28
784 pixel

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

pixel in scala di grigi = [0,255]

Label «otto»



28 x 28
784 pixel

Test set

addestrare

Modello matematico

astrarre

Modello predittivo

$\hat{y} = \mathbf{x0}*w0 + \mathbf{x1}*w1 + \mathbf{x2}*w2 + \ldots$

classificatore OTTO

784 pixel

bias

label

	x0	X1	X2	X3	X4	X5	X6	…	…	…	X784	Y
	1	0	162	100	0	100	180	…	…	…	255	otto

784 pixel

bias

label

	x0	X1	X2	X3	X4	X5	X6	…	…	…	X784	Y
	1	0	162	100	0	100	180	…	…	…	255	otto
	1	0	158	110	0	100	188	…	…	…	23	otto
	1	240	149	101	0	111	255	…	…	...	210	otto
	1	88	95	59	39	76	255	…	…	…	208	otto
	1	100	128	99	50	109	22	…	…	…	108	otto
	1	200	208	100	22	123	99	…	…	…	109	otto
	1	200	47	120	100	200	0	…	…	…	110	otto
	1	130	200	255	255	255	200	…	…	…	189	otto
	1	0	150	99	88	29	120	…	…	…	200	otto
	1	130	100	200	120	130	120	…	…	…	210	otto
	…	…	…	…	…	…	…	…	…	…	…	…

60K

TRAINING SET

X = 785 VARIABILI di ingresso

label

	x0	X1	X2	X3	X4	X5	X6	…	…	…	X784	Y
	1	0	162	100	0	100	180	…	…	…	255	otto
	1	0	158	110	0	100	188	…	…	…	23	otto
	1	240	149	101	0	111	255	…	…	...	210	otto
	1	88	95	59	39	76	255	…	…	…	208	otto
	1	100	128	99	50	109	22	…	…	…	108	otto
	1	200	208	100	22	123	99	…	…	…	109	otto
	1	200	47	120	100	200	0	…	…	…	110	otto
	1	130	200	255	255	255	200	…	…	…	189	otto
	1	0	150	99	88	29	120	…	…	…	200	otto
	1	130	100	200	120	130	120	…	…	…	210	otto
	…	…	…	…	…	…	…	…	…	…	…	…

60K

TRAINING SET

785 VARIABILI di ingresso

	x0	X1	X2	X3	X4	X5	X6	…	…	…	X784
	1	0	162	100	0	100	180	…	…	…	255
	1	0	158	110	0	100	188	…	…	…	23
	1	240	149	101	0	111	255	…	…	...	210
	1	88	95	59	39	76	255	…	…	…	208
	1	100	128	99	50	109	22	…	…	…	108
	1	200	208	100	22	123	99	…	…	…	109
	1	200	47	120	100	200	0	…	…	…	110
	1	130	200	255	255	255	200	…	…	…	189
	1	0	150	99	88	29	120	…	…	…	200
	1	130	100	200	120	130	120	…	…	…	210
	…	…	…	…	…	…	…	…	…	…	…

X

*

pesi
w0
w1
w2
…
…
…
w784

W

=

$\hat{Y}$

prediction

w0***x0** + w1***x1** + w2***x2** …

numero continuo

X = 785 VARIABILI di ingresso

	x0	X1	X2	X3	X4	X5	X6	…	…	…	X784	label Y
	1	0	162	100	0	100	180	…	…	…	255	otto
	1	0	158	110	0	100	188	…	…	…	23	otto
	1	240	149	101	0	111	255	…	…	...	210	otto
	1	88	95	59	39	76	255	…	…	…	208	otto
	1	100	128	99	50	109	22	…	…	…	108	otto
	1	200	208	100	22	123	99	…	…	…	109	otto
	1	200	47	120	100	200	0	…	…	…	110	otto
	1	130	200	255	255	255	200	…	…	…	189	otto
	1	0	150	99	88	29	120	…	…	…	200	otto
	1	130	100	200	120	130	120	…	…	…	210	otto
	…	…	…	…	…	…	…	…	…	…	…	…

TRAINING SET

REGRESSIONE LOGISTICA
POSITIVO = 1 o NEGATIVO = 0

REGRESSIONE LOGISTICA

	Classificazione a 2 classi	
$y \in \{0, 1\}$	**1 o classe positiva**	**0 o classe negativa**
Email	Spam	No spam
Tumore	Maligno	Benigno
Transazione bancaria	Fraudolenta	Sicura

X = 785 PARAMETRI

	x0	X1	X2	X3	X4	X5	X6	…	…	…	X784	label y
	1	0	162	100	0	100	180	…	…	…	255	otto
	1	0	158	110	0	100	188	…	…	…	23	otto
	1	240	149	101	0	111	255	…	…	...	210	otto
	1	88	95	59	39	76	255	…	…	…	208	otto
	1	100	128	99	50	109	22	…	…	…	108	otto
	1	200	208	100	22	123	99	…	…	…	109	otto
	1	200	47	120	100	200	0	…	…	…	110	otto
	1	130	200	255	255	255	200	…	…	…	189	otto
	1	0	150	99	88	29	120	…	…	…	200	otto
	1	130	100	200	120	130	120	…	…	…	210	otto
	…	…	…	…	…	…	…	…	…	…	…	…

TRAINING SET

X = 785 PARAMETRI

	x0	X1	X2	X3	X4	X5	X6	…	…	…	X784	label y
	1	0	162	100	0	100	180	…	…	…	255	1
	1	0	158	110	0	100	188	…	…	…	23	1
	1	240	149	101	0	111	255	…	…	...	210	1
	1	88	95	59	39	76	255	…	…	…	208	1
	1	100	128	99	50	109	22	…	…	…	108	1
	1	200	208	100	22	123	99	…	…	…	109	1
	1	200	47	120	100	200	0	…	…	…	110	1
	1	130	200	255	255	255	200	…	…	…	189	1
	1	0	150	99	88	29	120	…	…	…	200	1
	1	130	100	200	120	130	120	…	…	…	210	1
	…	…	…	…	…	…	…	…	…	…	…	0

TRAINING SET

Funzione di loss

prediction(X,W) :

return w0***x0** + w1***x1** + w2***x2** + w3***x3** + …

Funzione di loss

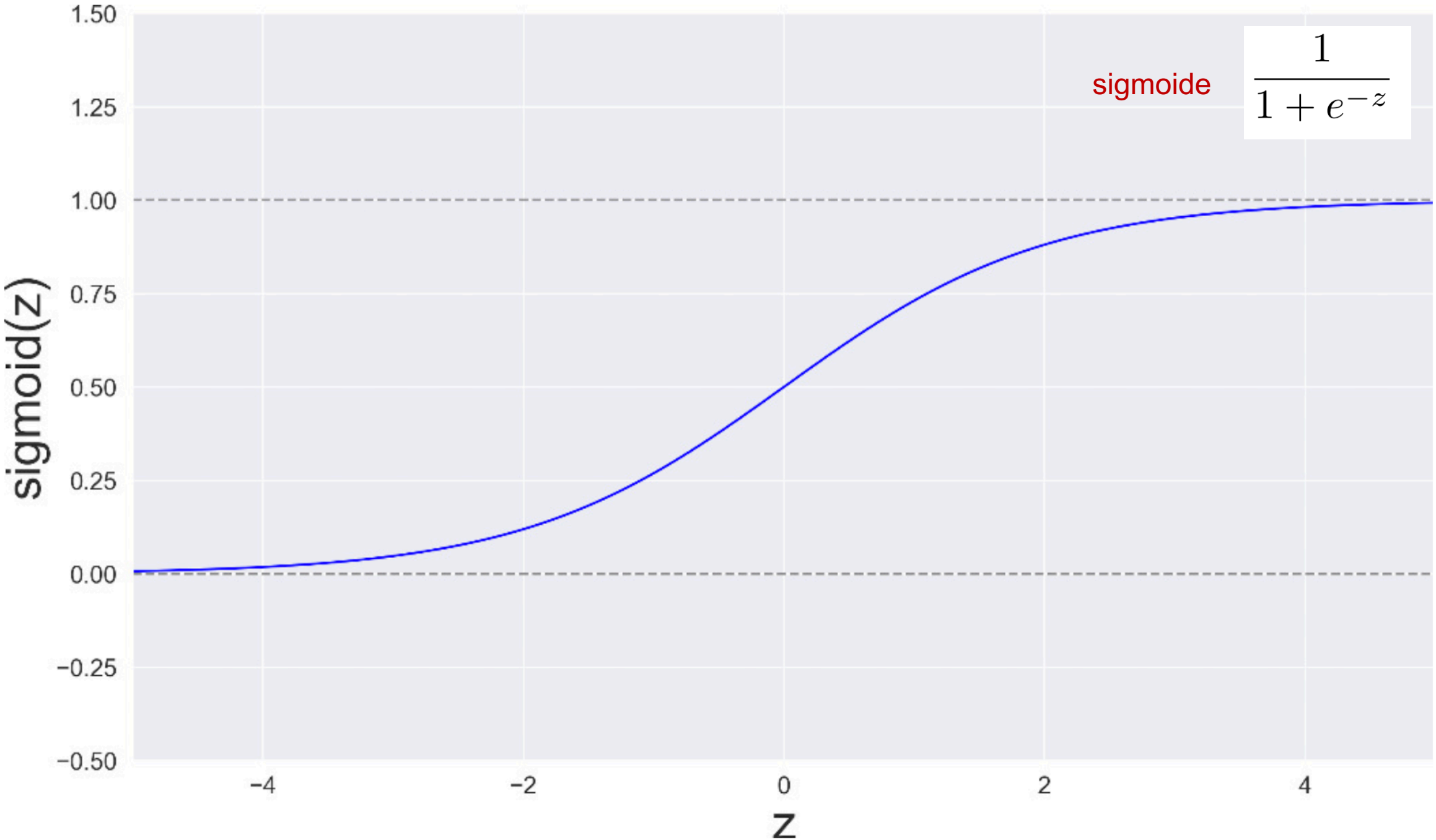
]0,1[*prediction(X,W)* :

return wrapper(w0***x0** + w1***x1** + w2***x2** + w3***x3** + …)

Funzione di loss

]0,1[*prediction(X,W)* :

return sigmoide(w0***x0** + w1***x1** + w2***x2** + w3***x3** + …)


sigmoide
$\frac{1}{1+e^{-z}}$
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
sigmoid(z)
−4
−2
0
2
4
z

Funzione di loss

prediction(X,W) :

return sigmoide(w0***x0** + w1***x1** + w2***x2** + w3***x3** + …)

]0,1[

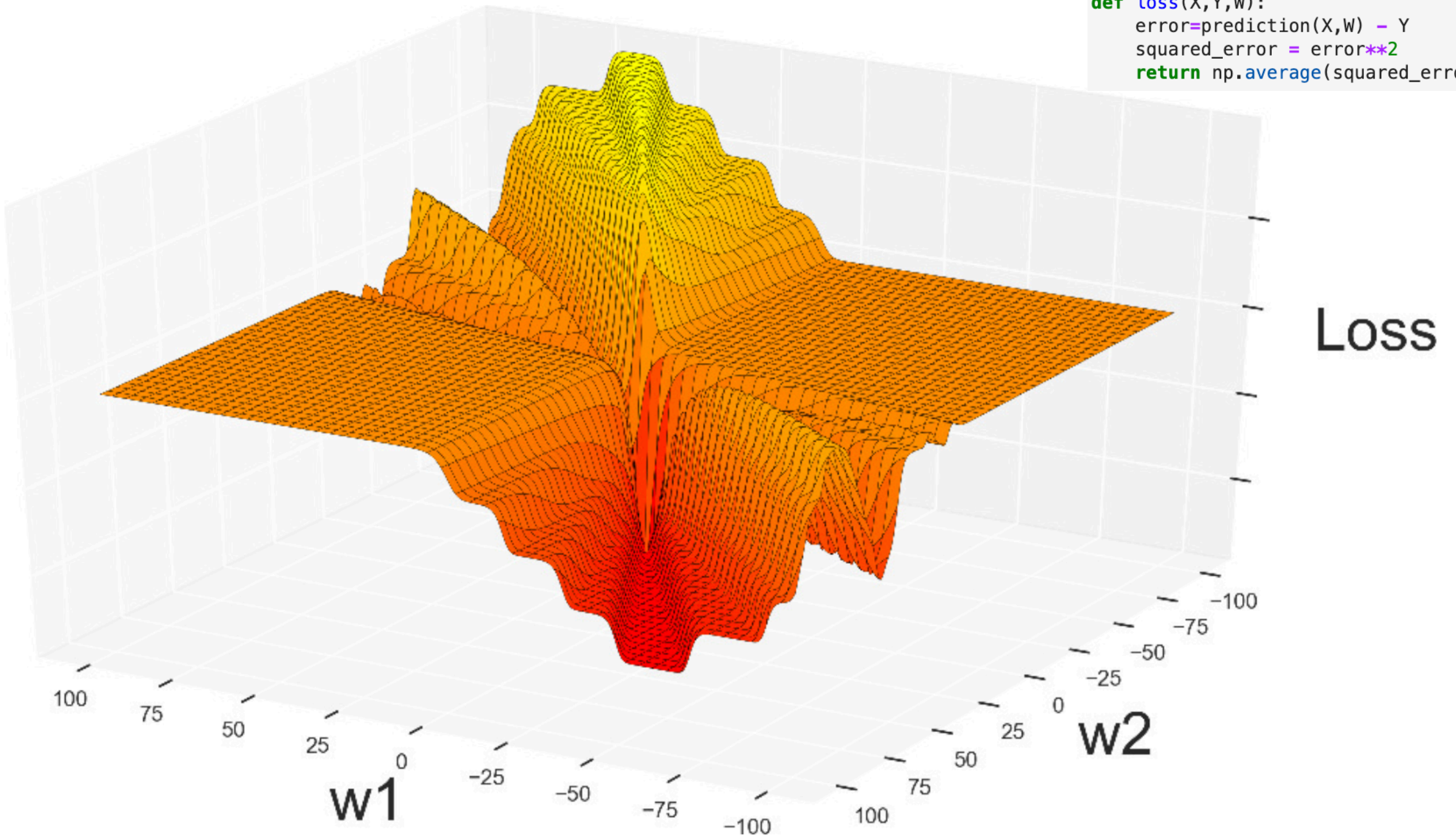
loss (X,W,Y) :

error = prediction(X,W) – Y

squared_error = error ** 2

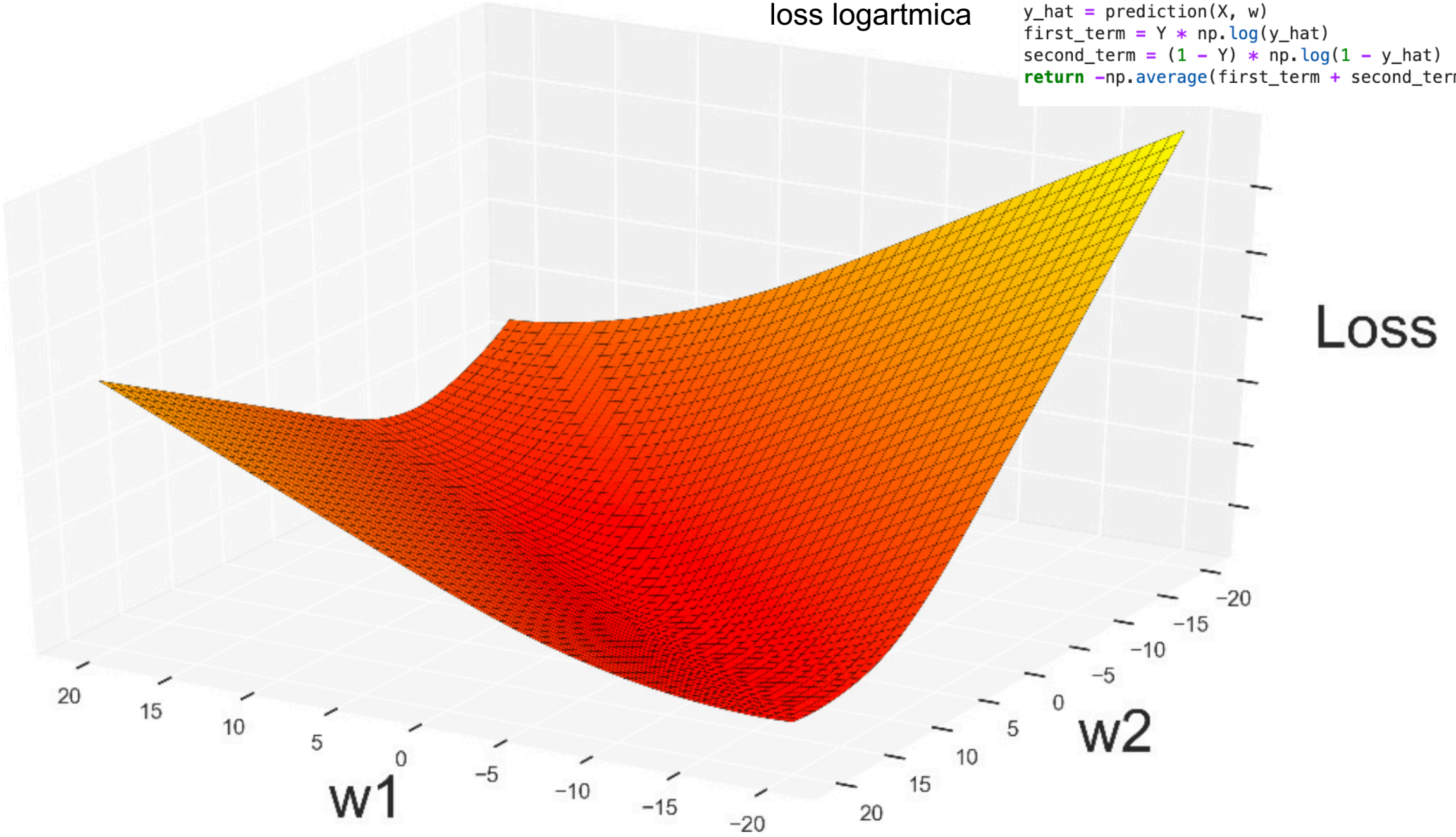
return media(error_squared)

```python
def loss(X,Y,W):
    error=prediction(X,W) - Y
    squared_error = error**2
    return np.average(squared_error)
```



loss logartmica

```
def loss(X, Y, w):
y_hat = prediction(X, w)
first_term = Y * np.log(y_hat)
second_term = (1 - Y) * np.log(1 - y_hat)
return -np.average(first_term + second_term)
```



MACHINE LEARNING:
riconoscere la cifra «otto»

TRAINING SET

Mnist (60k)

Gli «**otto**» gli etichetto con **classe positiva 1**
tutti gli altri con la **classe negativa 0**

addestro

Modello matematico

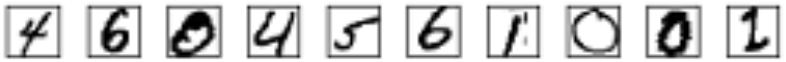
Regressione logistica

- $\hat{y}$ = sigmoide(X*W)
- loss_log(X,Y,W)

astraggo il ***Concetto***

W
Vettore pesi

TEST SET



predire

Modello predittivo

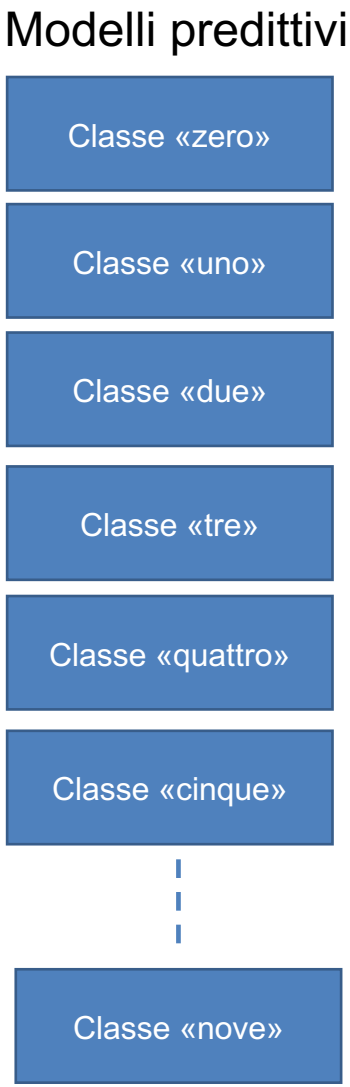
assegno

[0,1] %

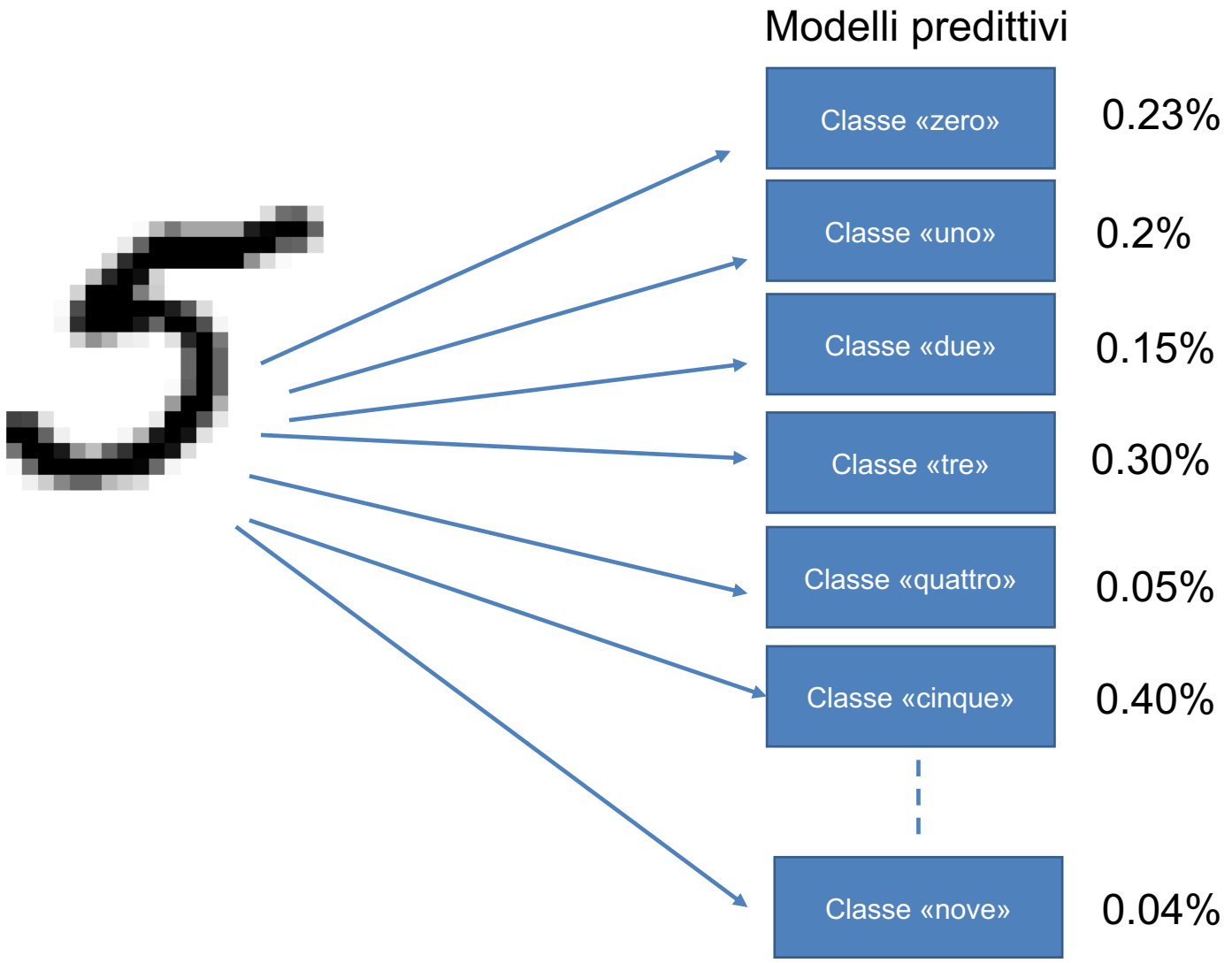
CLASSE «otto»

$\hat{y}$ = sigmoide(X*W)

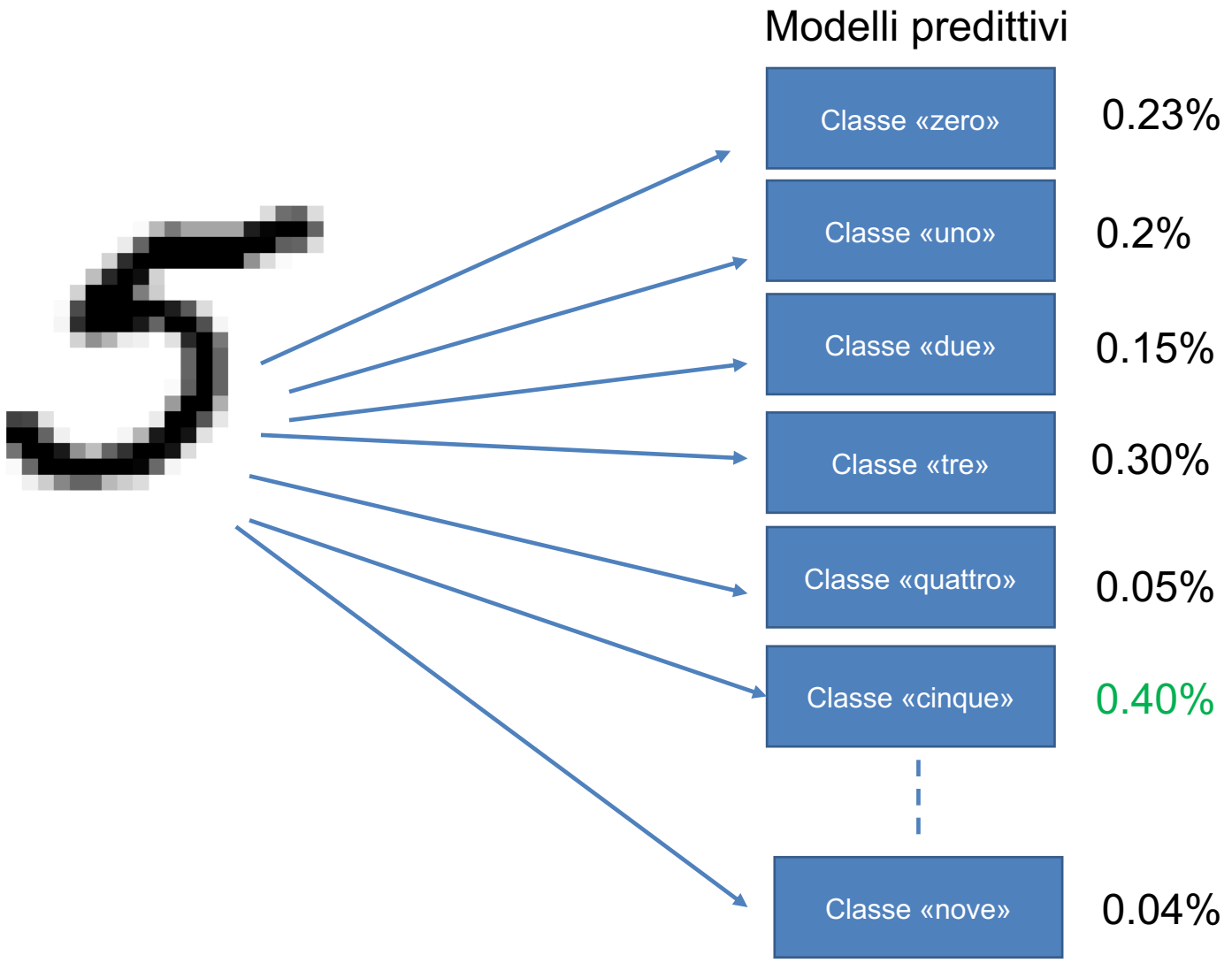
MACHINE LEARNING: riconoscere tutte le cifre



MACHINE LEARNING: riconoscere tutte le cifre



MACHINE LEARNING: riconoscere tutte le cifre



Come costruisco un modello predittivo per tutte le cifre?

X

X0	X1	X2	…	X784	LABEL	Y
1	100	200			tre	0
1	120	200			**otto**	1
1	200	100			cinque	0
1	200	76			zero	0
1	130	100			**otto**	1
…	…	…		…	…	…

TRAINING SET

Come costruisco un modello predittivo per tutte le cifre?

X

X0	X1	X2	…	X784	LABEL	Y									
						0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	100	200			tre	0	0	0	1	0	0	0	0	0	0
1	120	200			otto	0	0	0	0	0	0	0	0	1	0
1	200	100			cinque	0	0	0	0	0	1	0	0	0	0
1	200	76			zero	1	0	0	0	0	0	0	0	0	0
1	130	100			otto	0	0	0	0	0	0	0	0	1	0
…	…	…		…	…	…									

TRAINING SET

one-hot encoding

Cosa mi restituirà il mio modello predittivo?

X

X0	X1	X2	…	X784	Ŷ (CLASSIFICATORE)									
					0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	2	100		209	0.3	0.3	0.5	1	0.1	0.3	0.2	0.3	0.4	0.2
1	0	200		102	0.2	0.3	0.2	0.2	0.8	0.2	0.4	0.6	0.2	0.8
1	210	100		208	0.2	0.1	0.3	0.9	0.1	0.3	0.1	0.3	0.3	0.1
1	10	76		183	0.23	0.33	0.45	0.68	0.1	0.3	0.52	0.3	0.43	0.2
1	130	100		209	0.4	0.59	0.8	0.1	0.2	0.3	0.3	0.1	0.1	0.3
…	…	…		…	…									

CODE…

$$\underset{\text{matrice}}{\mathbf{X}} * \underset{\text{vettore}}{\mathbf{W}} = \underset{\text{vettore}}{\hat{\mathbf{Y}}}$$

	x0	X1	X2	X3	X4	X5	X6	…	…	…	X784
	1	0	162	100	0	100	180	…	…	…	255
	1	0	158	110	0	100	188	…	…	…	23
	1	240	149	101	0	111	255	…	…	...	210
	1	88	95	59	39	76	255	…	…	…	208
	1	100	128	99	50	109	22	…	…	…	108
	1	200	208	100	22	123	99	…	…	…	109
	1	200	47	120	100	200	0	…	…	…	110
	1	130	200	255	255	255	200	…	…	…	189
	1	0	150	99	88	29	120	…	…	…	200
	1	130	100	200	120	130	120	…	…	…	210
	…	…	…	…	…	…	…	…	…	…	…

X

matrice

*

pesi
w0
w1
w2
…
…
…
w784

W

vettore

=

$\hat{Y}$

prediction

vettore

	x0	X1	X2	X3	X4	X5	X6	…	…	…	X784
	1	0	162	100	0	100	180	…	…	…	255
	1	0	158	110	0	100	188	…	…	…	23
	1	240	149	101	0	111	255	…	…	...	210
	1	88	95	59	39	76	255	…	…	…	208
	1	100	128	99	50	109	22	…	…	…	108
	1	200	208	100	22	123	99	…	…	…	109
	1	200	47	120	100	200	0	…	…	…	110
	1	130	200	255	255	255	200	…	…	…	189
	1	0	150	99	88	29	120	…	…	…	200
	1	130	100	200	120	130	120	…	…	…	210
	…	…	…	…	…	…	…	…	…	…	…

X

matrice

*

pesi
w0
w1
w2
…
…
…
w784

W

vettore

=

$\hat{Y}$

prediction

matrice

	x0	X1	X2	X3	X4	X5	X6	…	…	…	X784
	1	0	162	100	0	100	180	…	…	…	255
	1	0	158	110	0	100	188	…	…	…	23
	1	240	149	101	0	111	255	…	…	...	210
	1	88	95	59	39	76	255	…	…	…	208
	1	100	128	99	50	109	22	…	…	…	108
	1	200	208	100	22	123	99	…	…	…	109
	1	200	47	120	100	200	0	…	…	…	110
	1	130	200	255	255	255	200	…	…	…	189
	1	0	150	99	88	29	120	…	…	…	200
	1	130	100	200	120	130	120	…	…	…	210
	…	…	…	…	…	…	…	…	…	…	…

X

matrice
(m, n)

*

pesi
w0
w1
w2
…
…
…
w784

W

vettore

=

$\hat{Y}$

prediction

matrice
(k, n)

	x0	X1	X2	X3	X4	X5	X6	…	…	…	X784
	1	0	162	100	0	100	180	…	…	…	255
	1	0	158	110	0	100	188	…	…	…	23
	1	240	149	101	0	111	255	…	…	...	210
	1	88	95	59	39	76	255	…	…	…	208
	1	100	128	99	50	109	22	…	…	…	108
	1	200	208	100	22	123	99	…	…	…	109
	1	200	47	120	100	200	0	…	…	…	110
	1	130	200	255	255	255	200	…	…	…	189
	1	0	150	99	88	29	120	…	…	…	200
	1	130	100	200	120	130	120	…	…	…	210
	…	…	…	…	…	…	…	…	…	…	…

X

matrice
(m, n)

*

W1	…	Wk
w0		..
w1		
w2		
…		
…		
…		
w784		

W
matrice

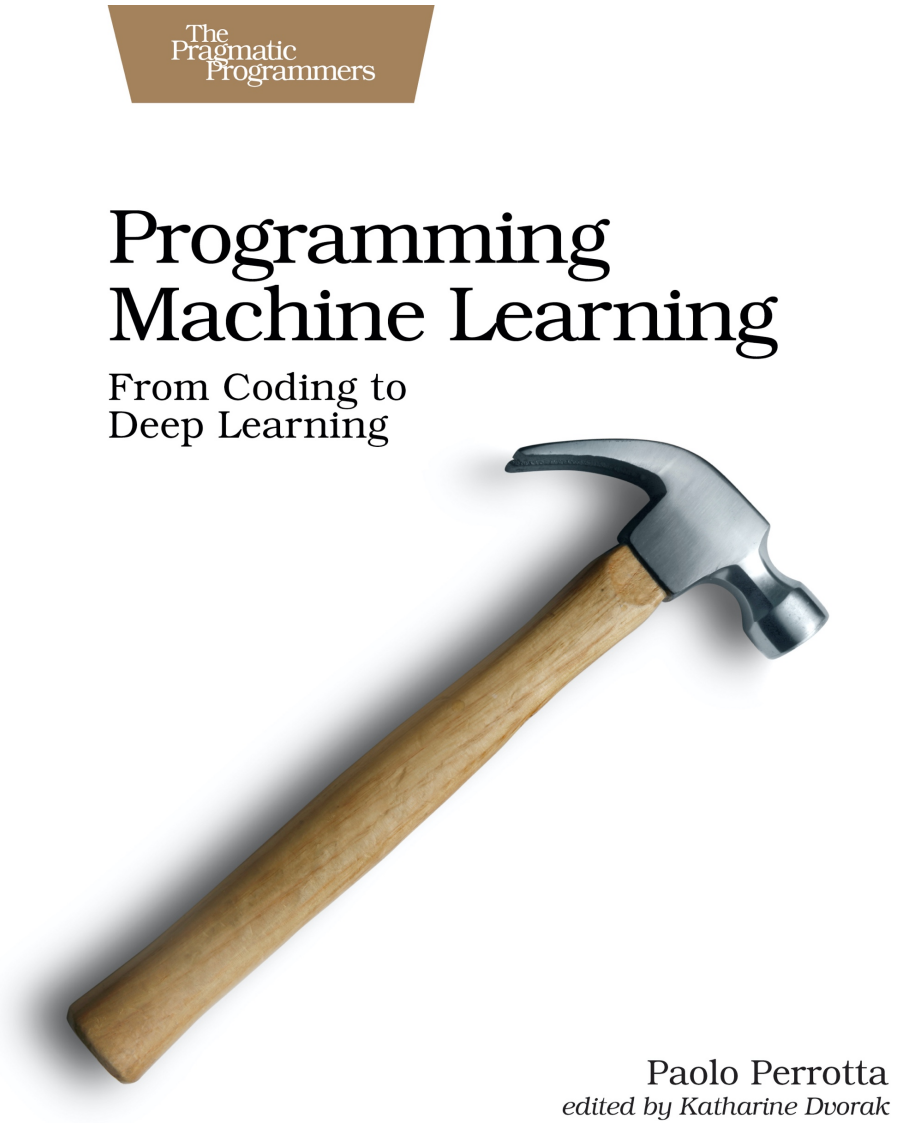
=

$\hat{Y}$

prediction

matrice
(k, n)

Riferimenti e Risorse Utili



Riferimenti e Risorse Utili

https://course.fast.ai/

COMMUNITY E COMPETIZIONI MACHINE LEARNING

https://www.kaggle.com/