II - Linear and Logistic Regression

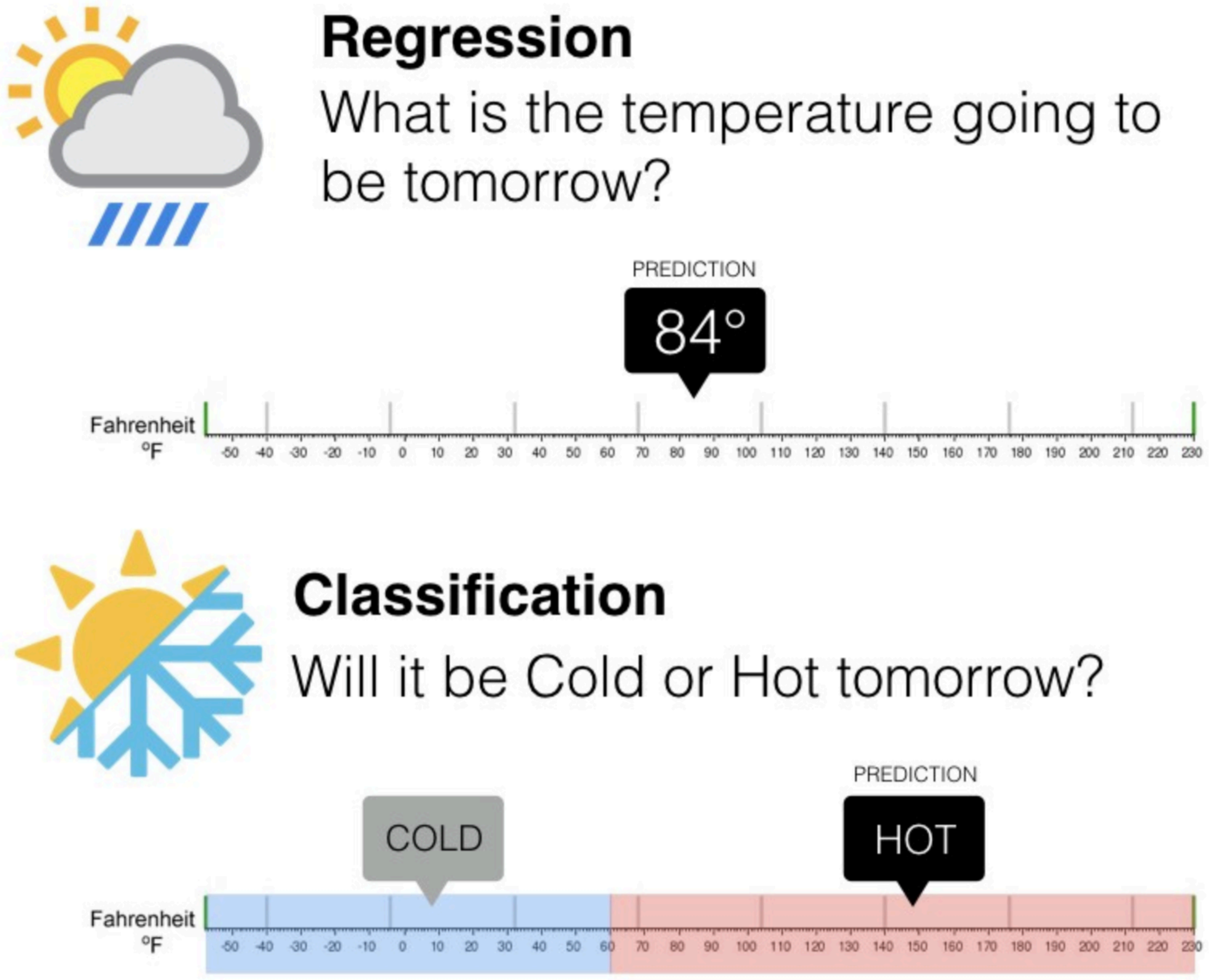
Formalism & Inference function

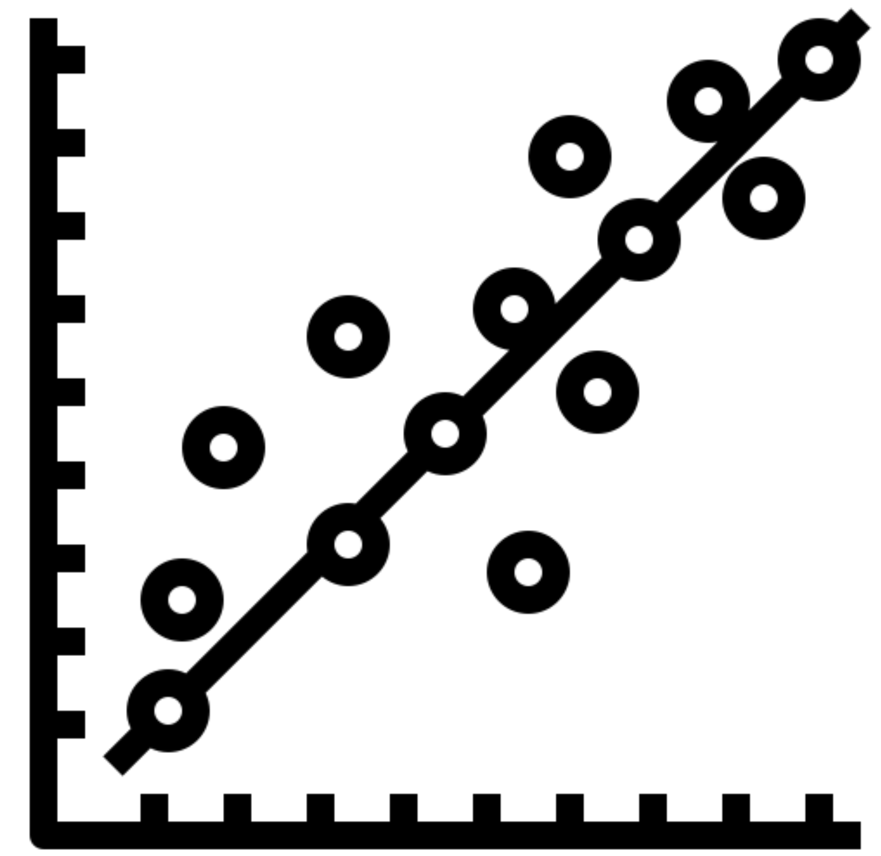
Loss function

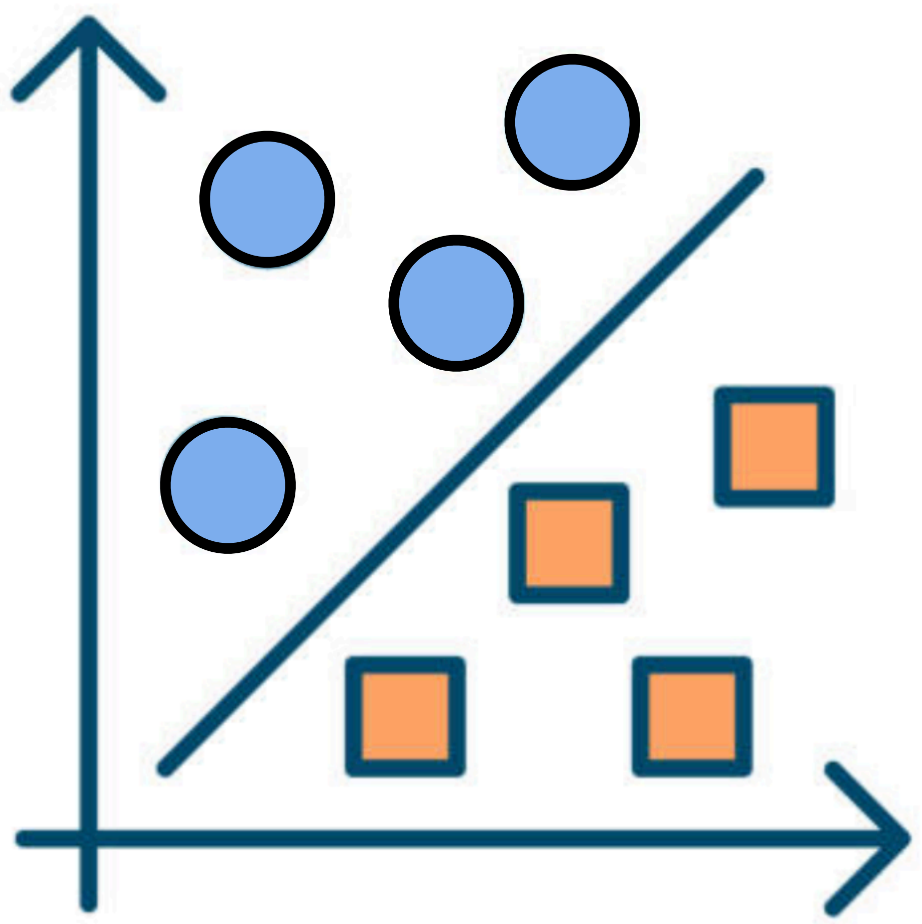
Algorithms

youssef.ait-el-mahjoub@efrei.fr

Regression Vs Classification







Source: Pintrest

Linear Regression

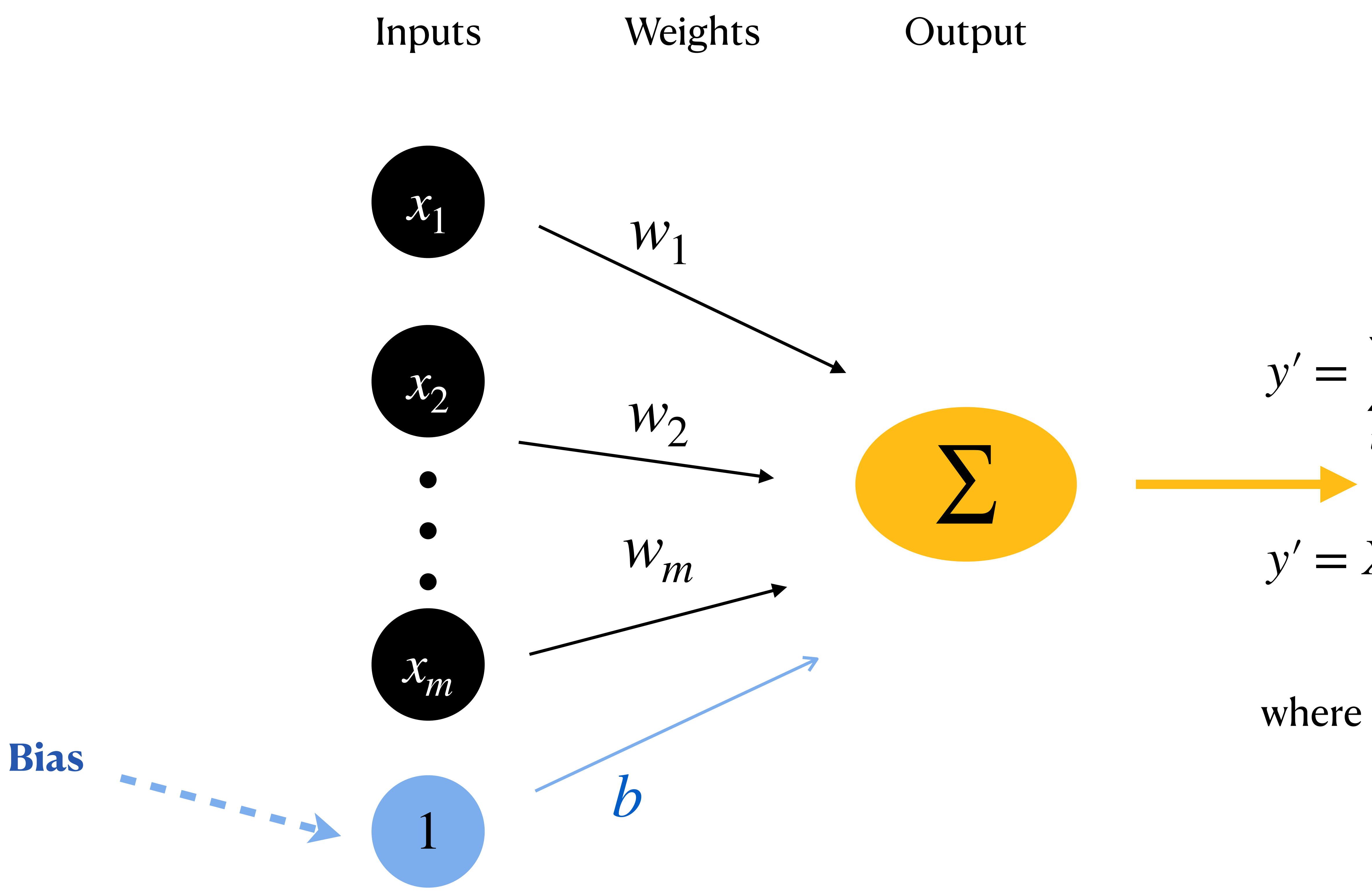
Formalism

- One of fundamental methods in ***superviserd Machine Learning***.
- Used to ***predict*** continous values.
- Predict the value of a *dependent variable* based on the values of another *independent variables*.

Linear Regression

Formalism

- One of fundamental methods in ***superviserd Machine Learning***.
- Used to ***predict*** continous values.
- Predict the value of a *dependent variable* based on the values of another *independent variables*.



$$y' = \sum_{i=1}^{m} x_i w_i + b$$

$$y' = X \,.\, W + b$$ (Matrix version)

where : X.W is a scalar product for each row in X by vector W

Linear Regression

Formalism

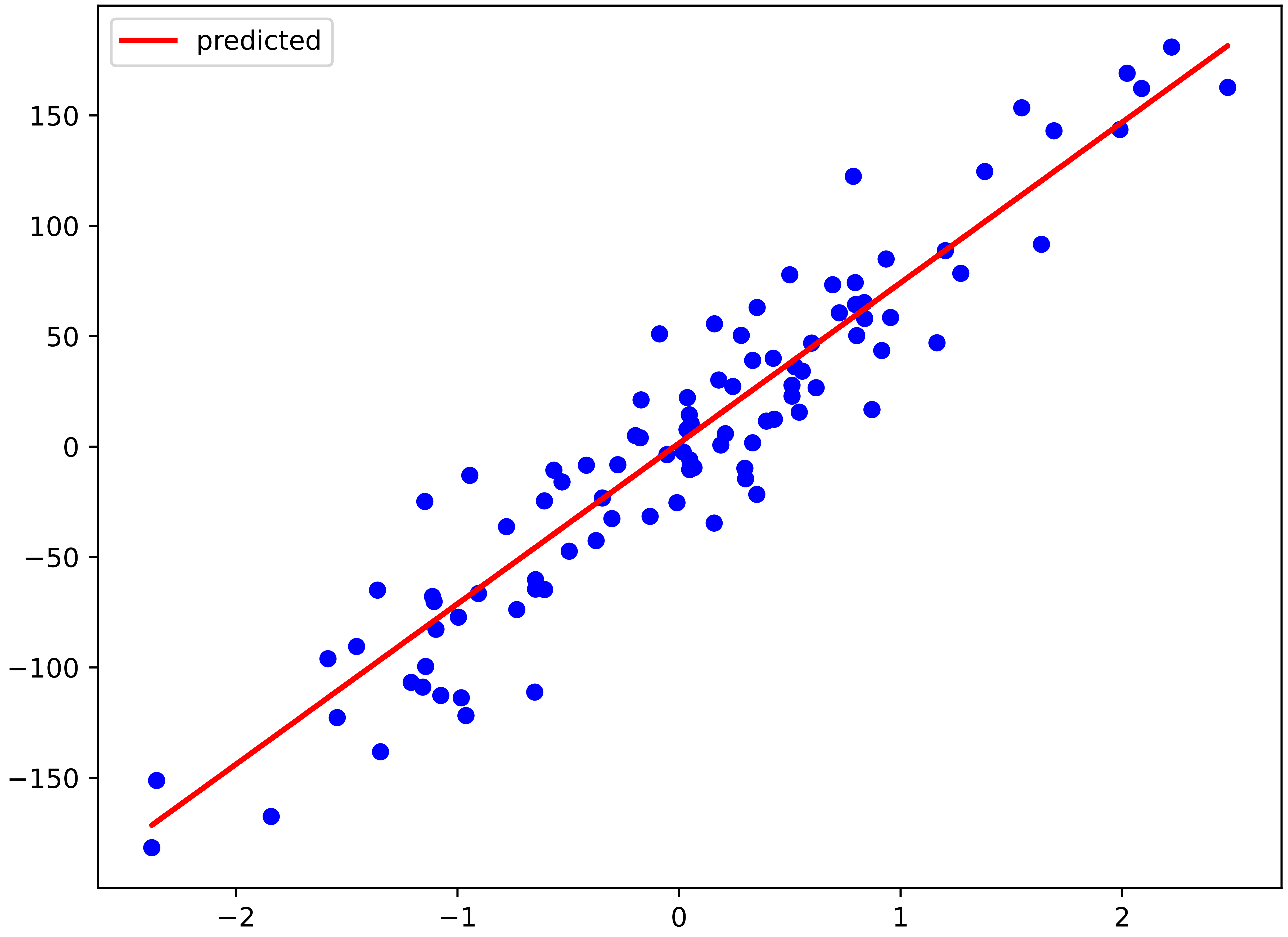
- One of fundamental methods in ***superviserd Machine Learning***.
- Used to ***predict*** continous values.
- Predict the value of a *dependent variable* based on the values of another *independent variables.*

★ General formulation, *the inference function* : $y = X \,.\, W + b$

- y is the dependent variable : continous values we want to predict
- X represent a data set of n samples and m features (independent variables).
- W is a vector of weights (***the unknown, we want to learn***), of size $(1, m)$.
- b is a scalar value called the bias (***unknown value, we want to learn***).

★ The goal is to find vector of weights **W** and scalar term **b**

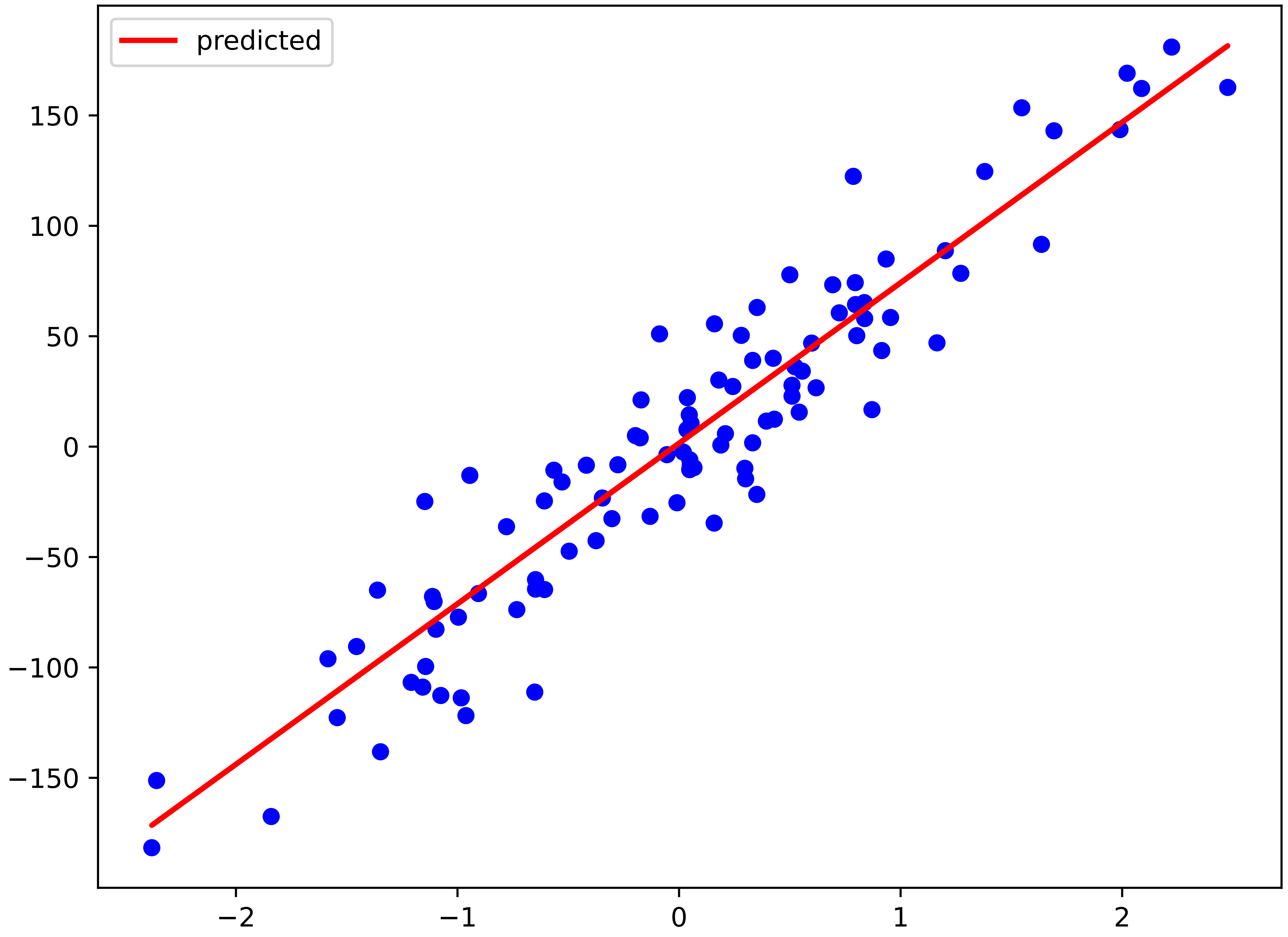
Linear Regression
Formalism



Linear Regression

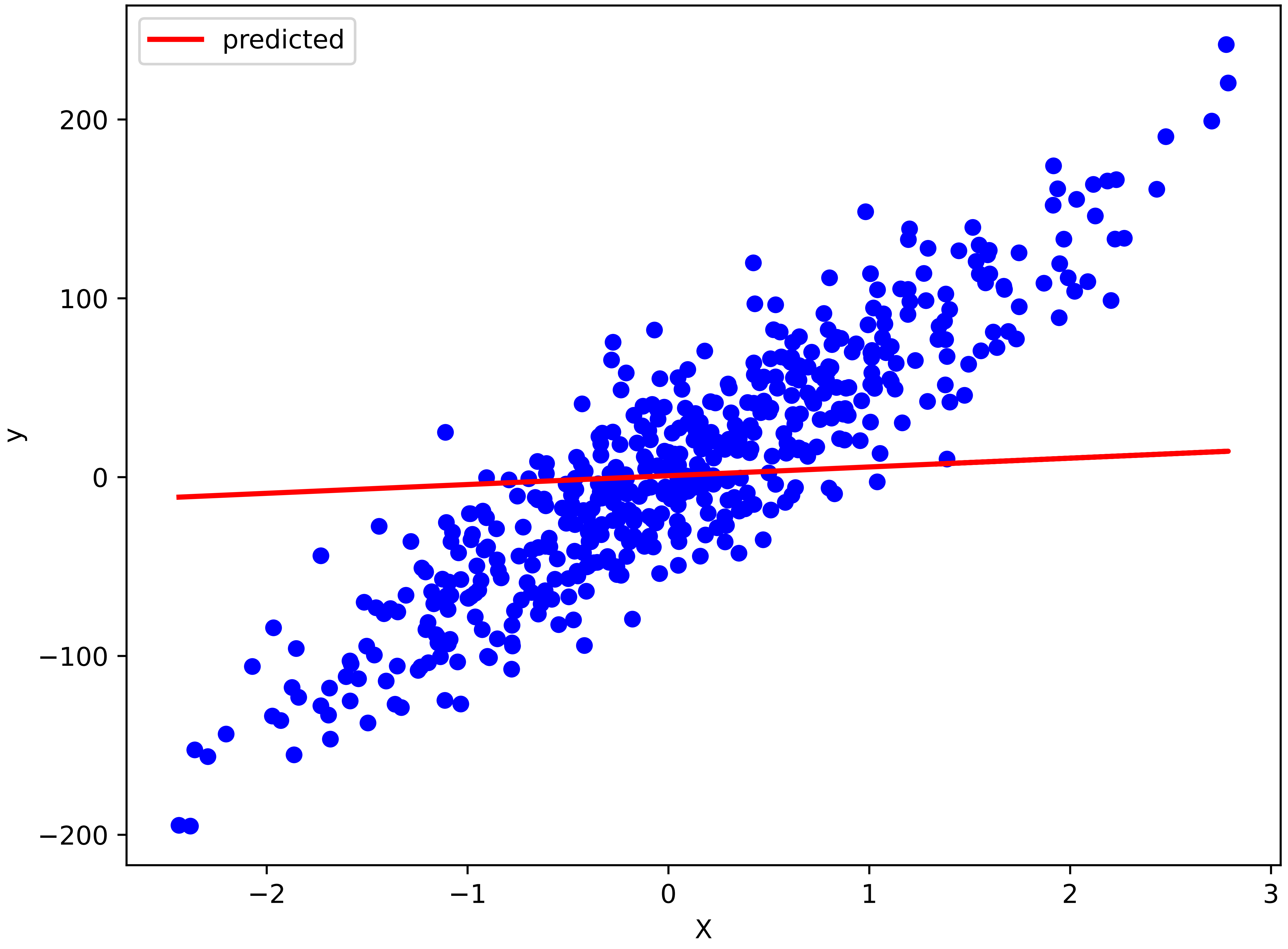
Formalism

- We want to find the equation fiting the ***predicted line*** to the ***data.***
- Example of data with only one weight (i.e. one feature 'w')
- Linear regression consists in finding this ***infinite red line*** (a line not a circle, arc, or other shape … otherwise we need non-linear methods).
- We call this line a ***regression model***.



Linear Regression
Formalism

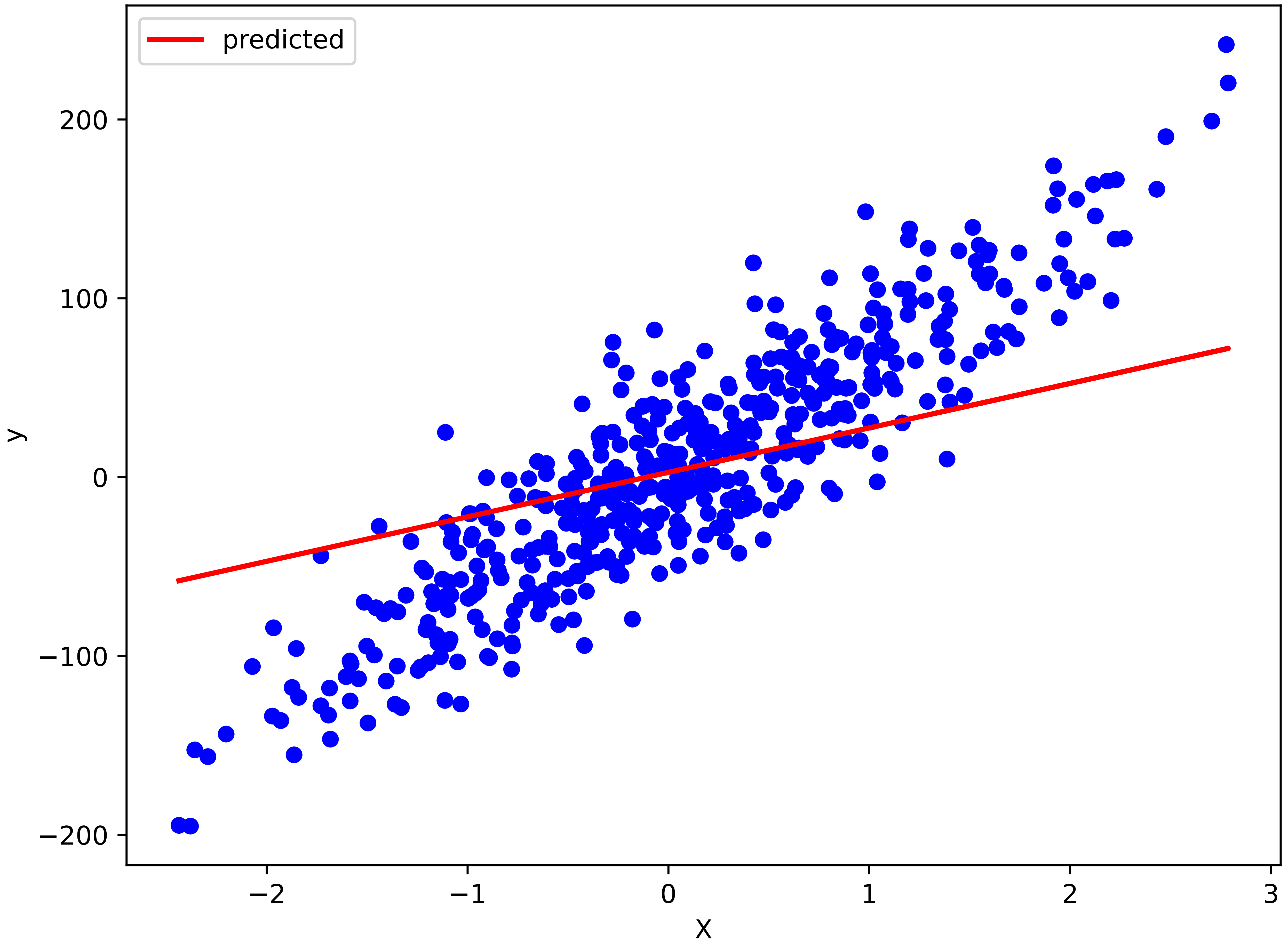
- Some value of weight 'w' and bias 'b'.
- These weights controls the shape of the line.



Linear Regression

Formalism

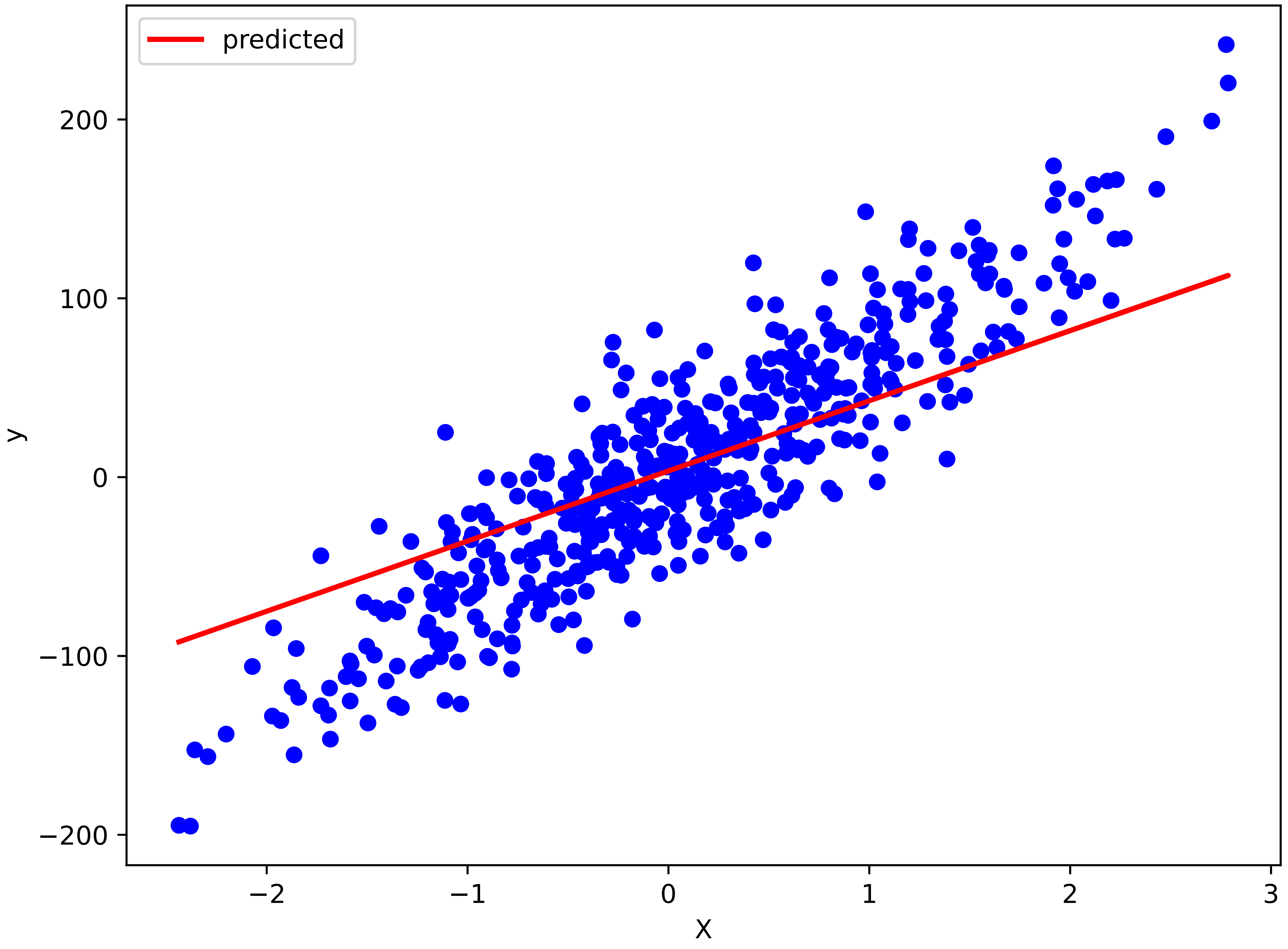
- Some value of weight 'w' and bias 'b'.
- These weights controls the shape of the line.



Linear Regression

Formalism

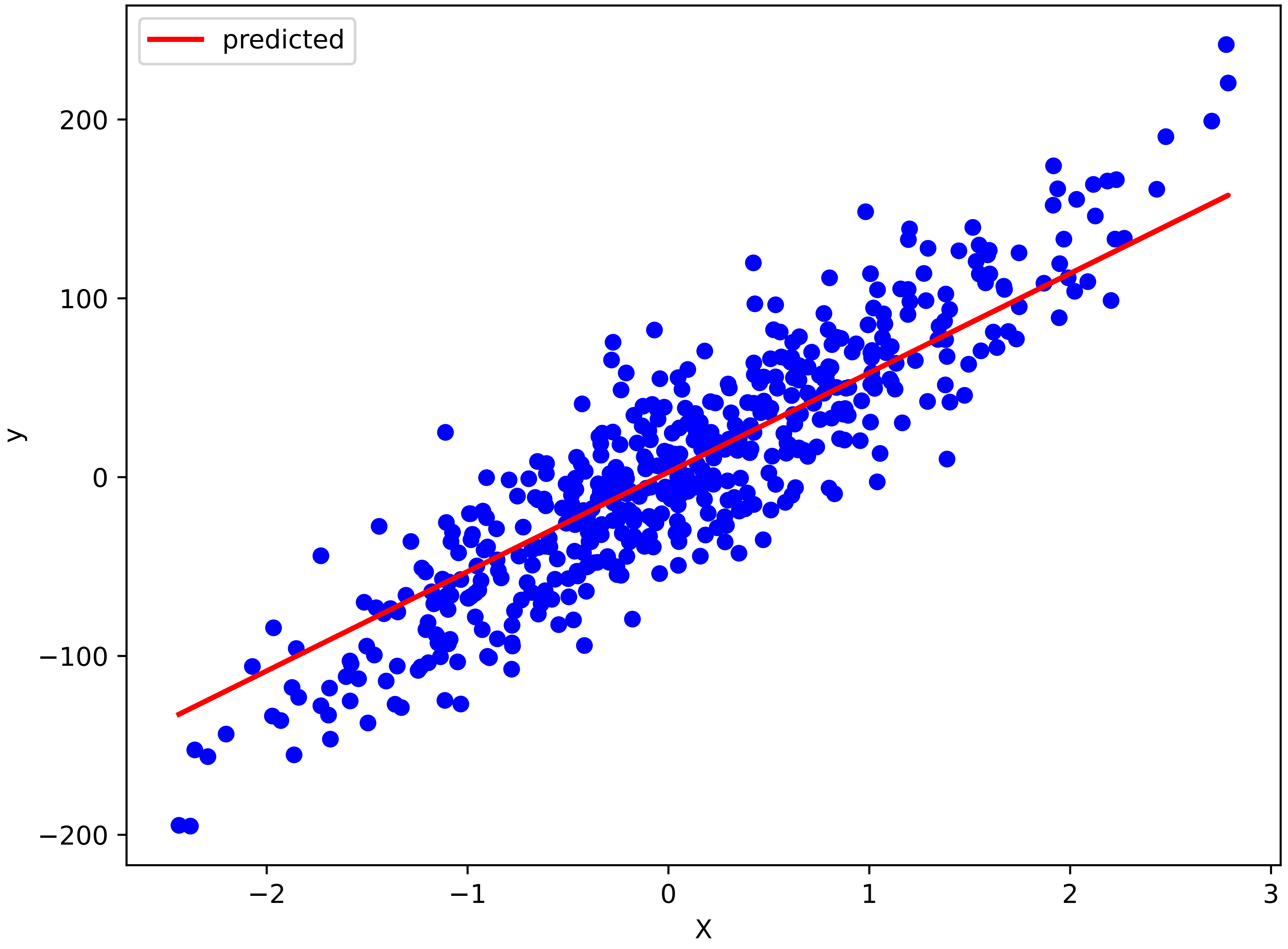
- Some value of weight 'w' and bias 'b'.
- These weights controls the shape of the line.



Linear Regression

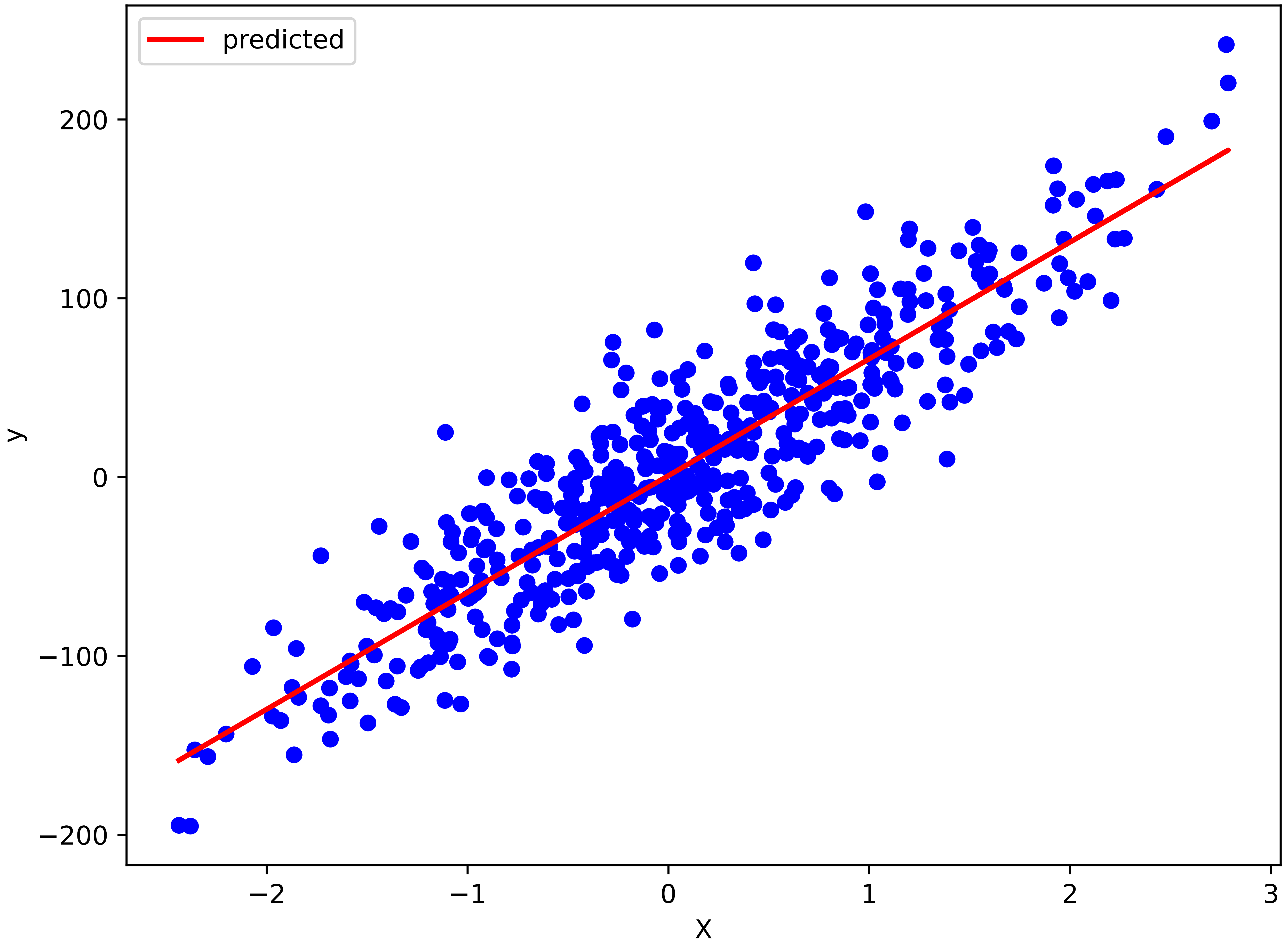
Formalism

- Some value of weight 'w' and bias 'b'.
- These weights controls the shape of the line.
- Almost !



Linear Regression
Formalism

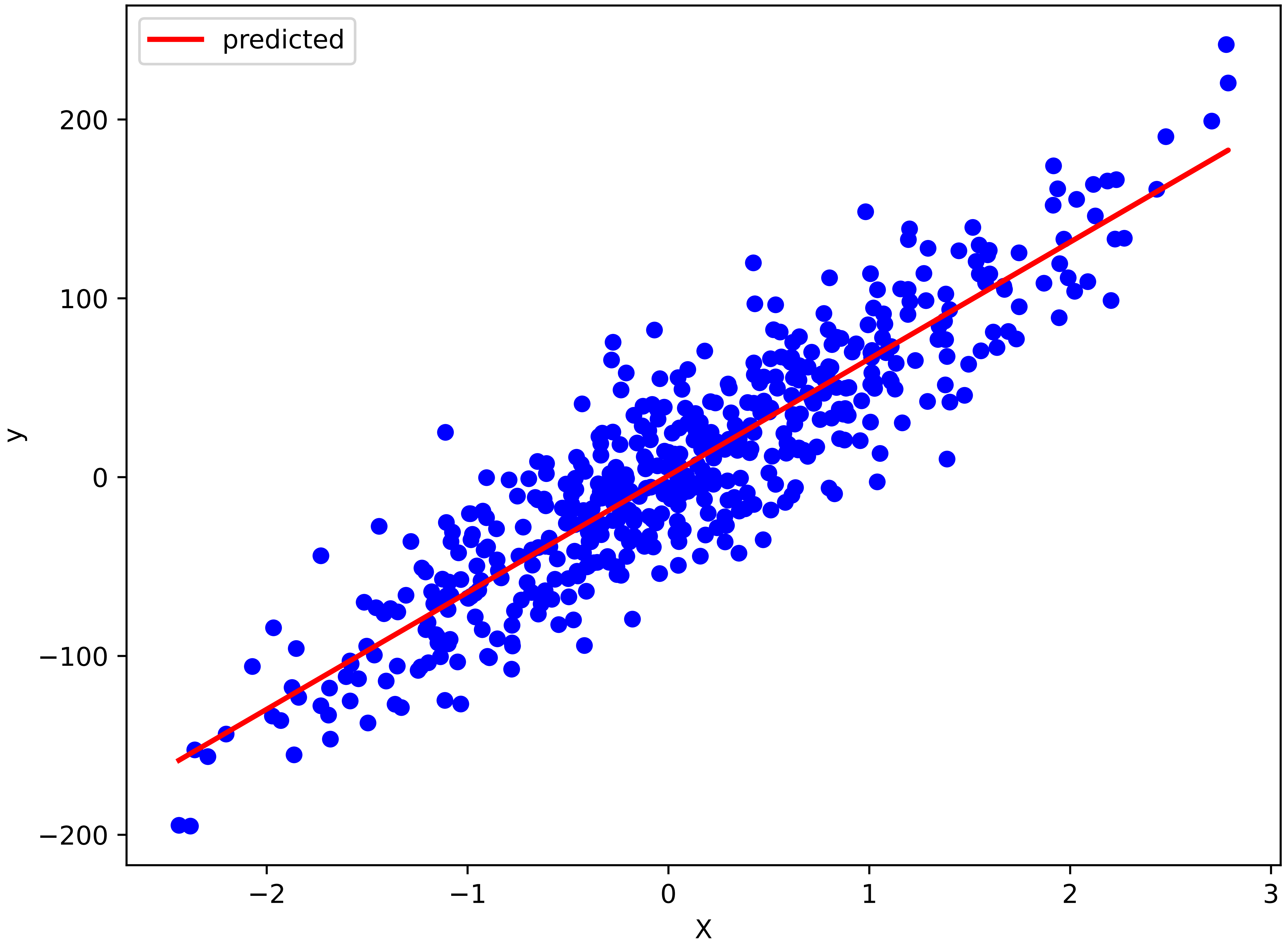
- Not easy to find the exact position of the line !
- We use an optimisation algorithm known as "**Gradient Descent Algorithm**".
- We can observe that the most fitting line is truly ***"close"*** to all the data points, especially when compared to the initial line (slide 6).



Linear Regression

Formalism

- Not easy to find the exact position of the line !
- We use an optimisation algorithm known as **"Gradient Descent Algorithm"**.
- We can observe that the most fitting line is truly ***"close"*** to all the data points, especially when compared to the initial line (slide 6).
- This closeness is mathematically quantified by ***a loss (or error) function***.
- The objective of GDA algorithm is to ***minimize*** this specified loss function.



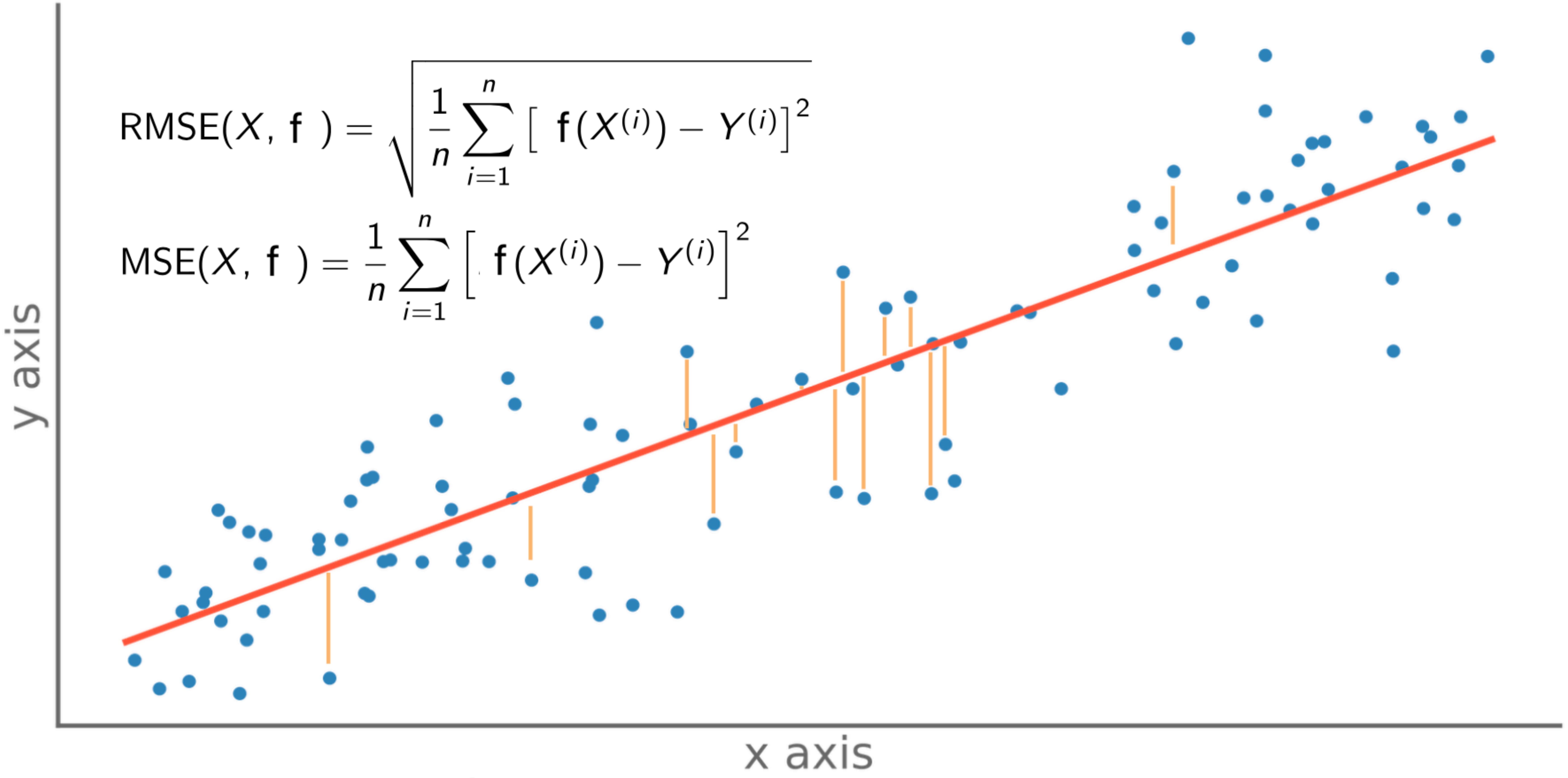
Linear Regression

Loss function

- Some of the most used error functions in LR :

✳ MSE : Mean Square Error function.

✳ RMSE : Root Mean Square Error function.



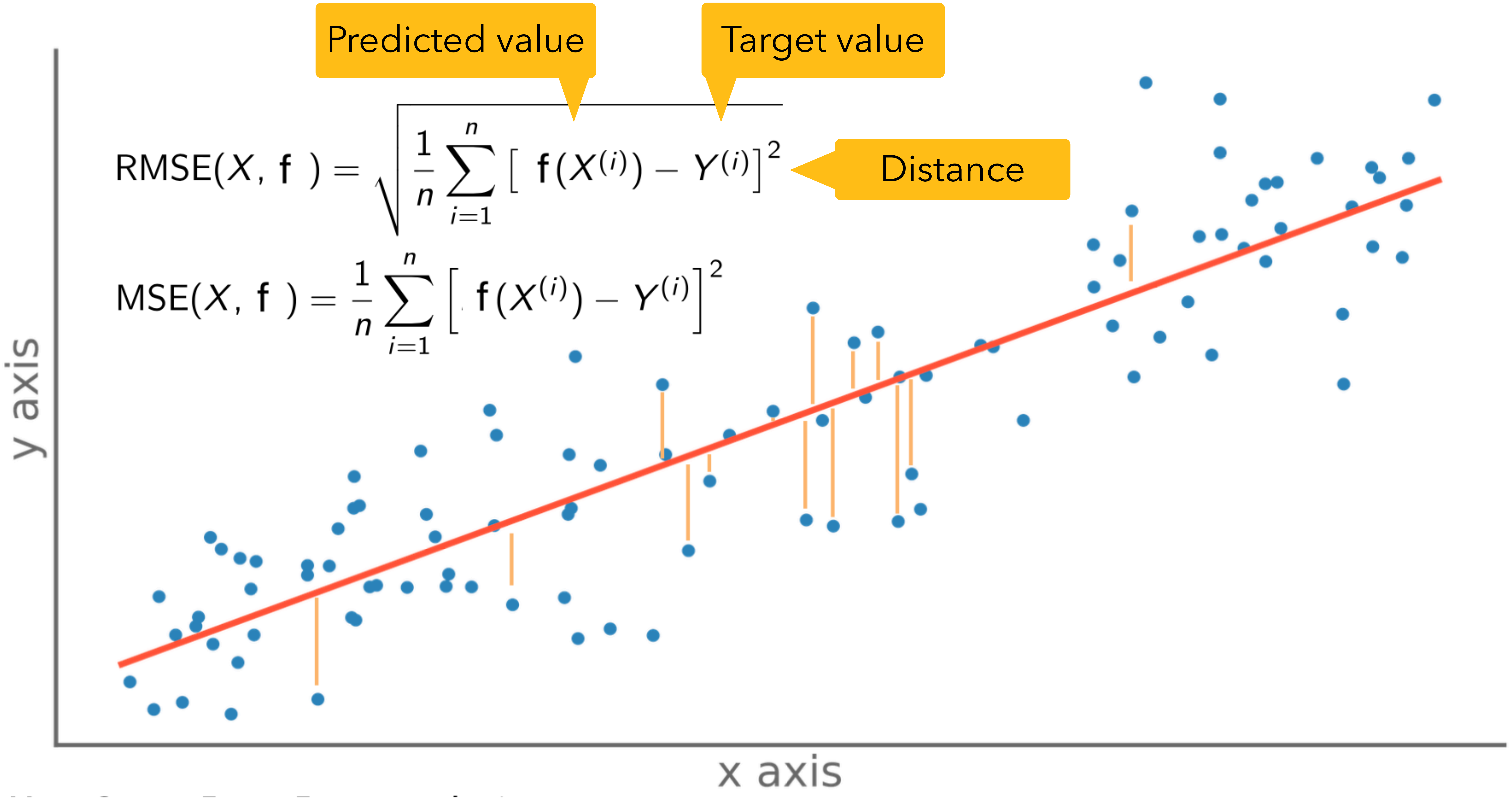
Linear Regression

Loss function

- Some of the most used error functions in LR :

✳ MSE : Mean Square Error function.

✳ RMSE : Root Mean Square Error function.



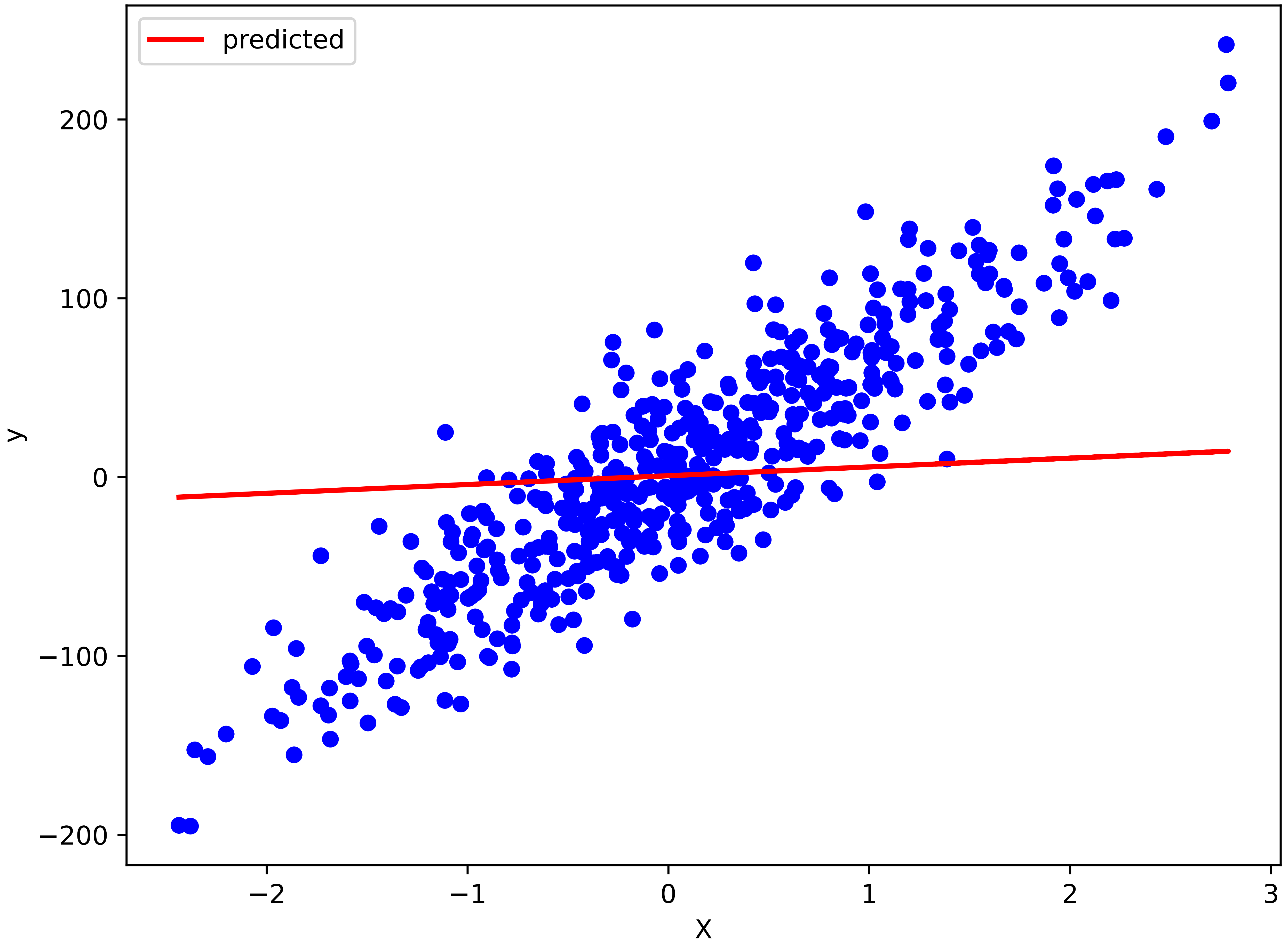
$$\mathrm{RMSE}(X, \mathrm{f}\ ) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[\ \mathrm{f}(X^{(i)}) - Y^{(i)}\right]^2}$$

$$\mathrm{MSE}(X, \mathrm{f}\ ) = \frac{1}{n}\sum_{i=1}^{n}\left[\ \mathrm{f}(X^{(i)}) - Y^{(i)}\right]^2$$

Linear Regression

Loss function

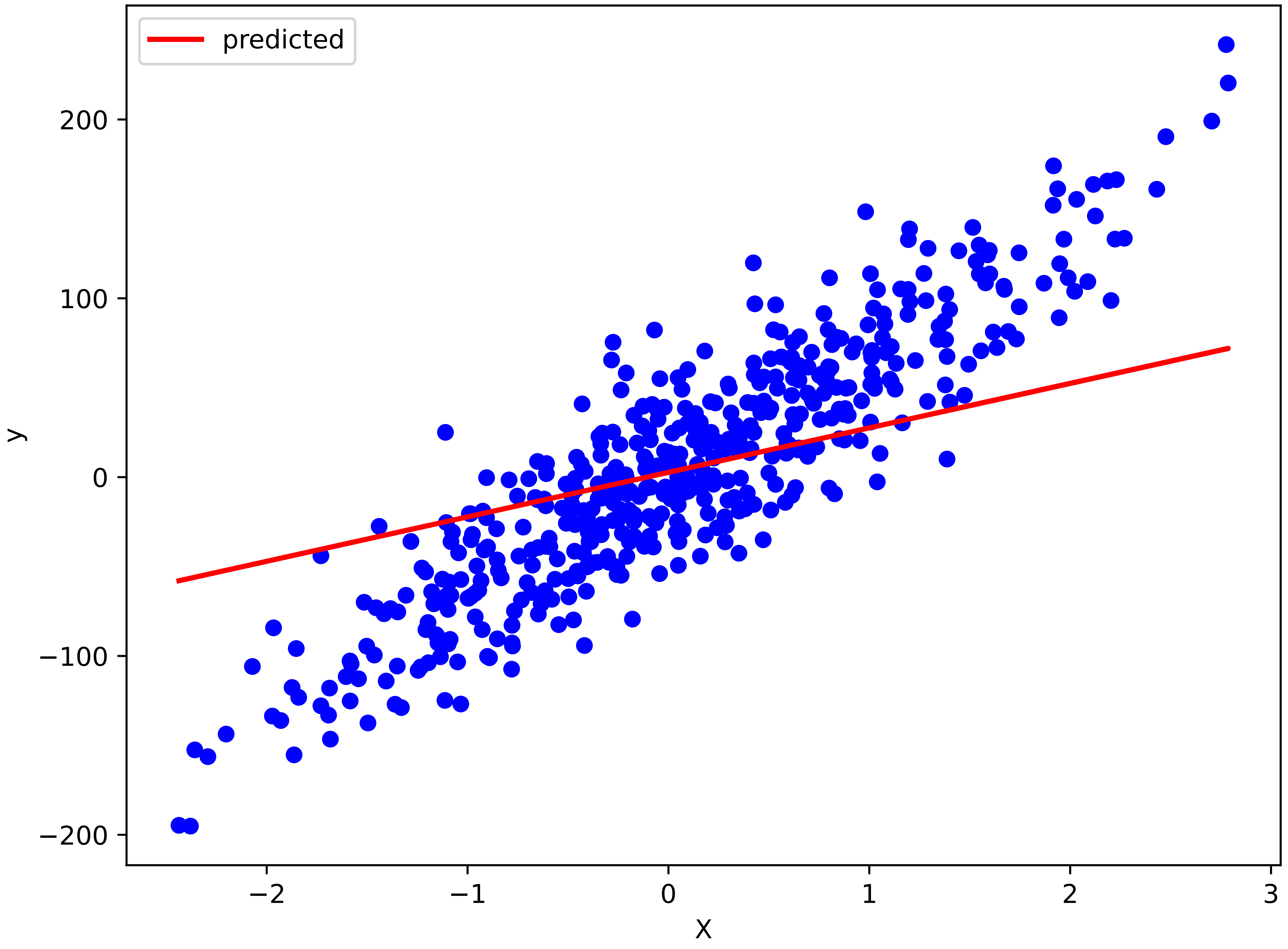
- Some value of weight 'w' and bias 'b'.
- These weights controls the shape of the line.
- MSE = 4600



Linear Regression

Loss function

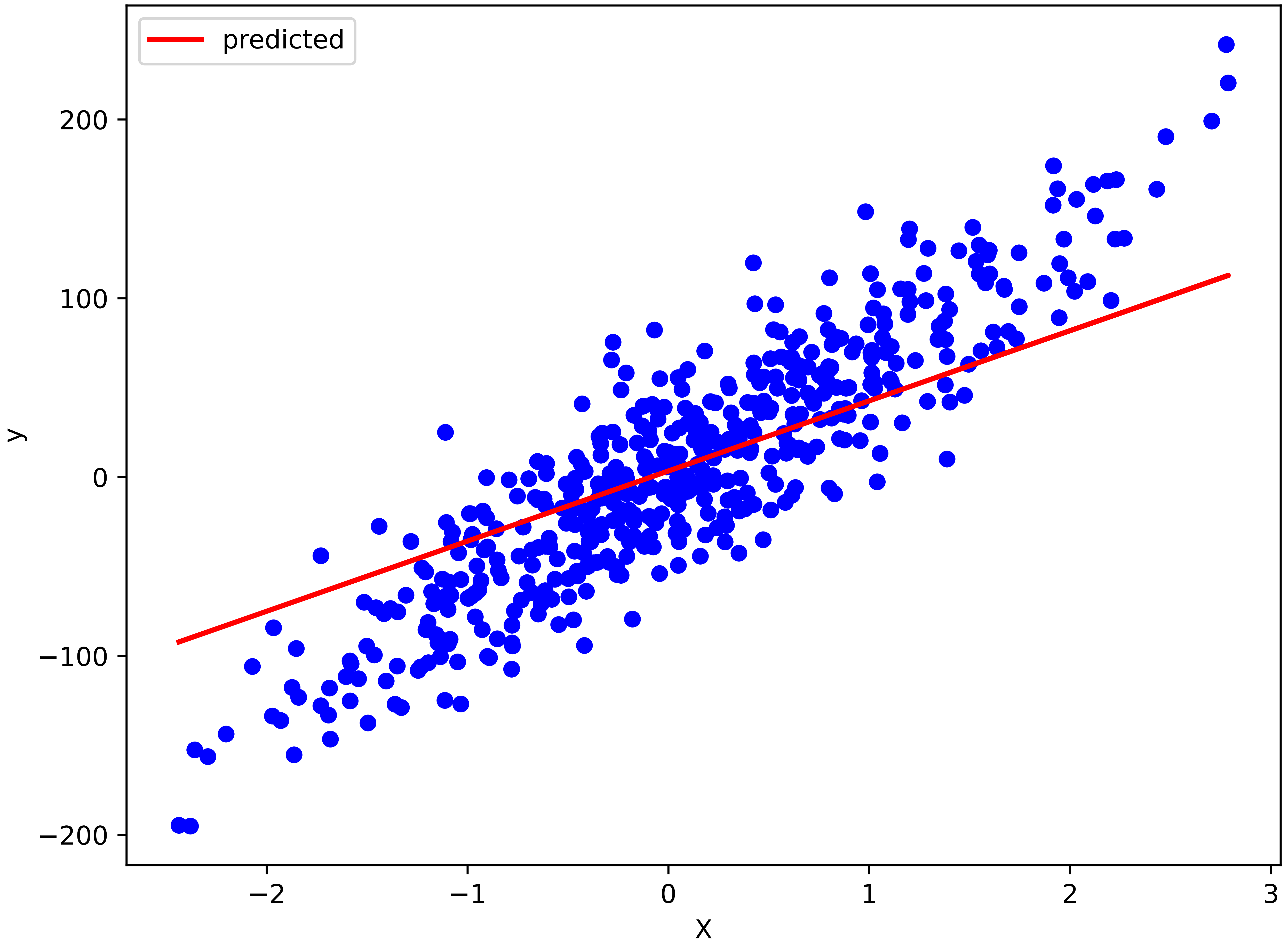
- Some value of weight 'w' and bias 'b'.
- These weights controls the shape of the line.
- MSE = 2495 ↓



Linear Regression

Loss function

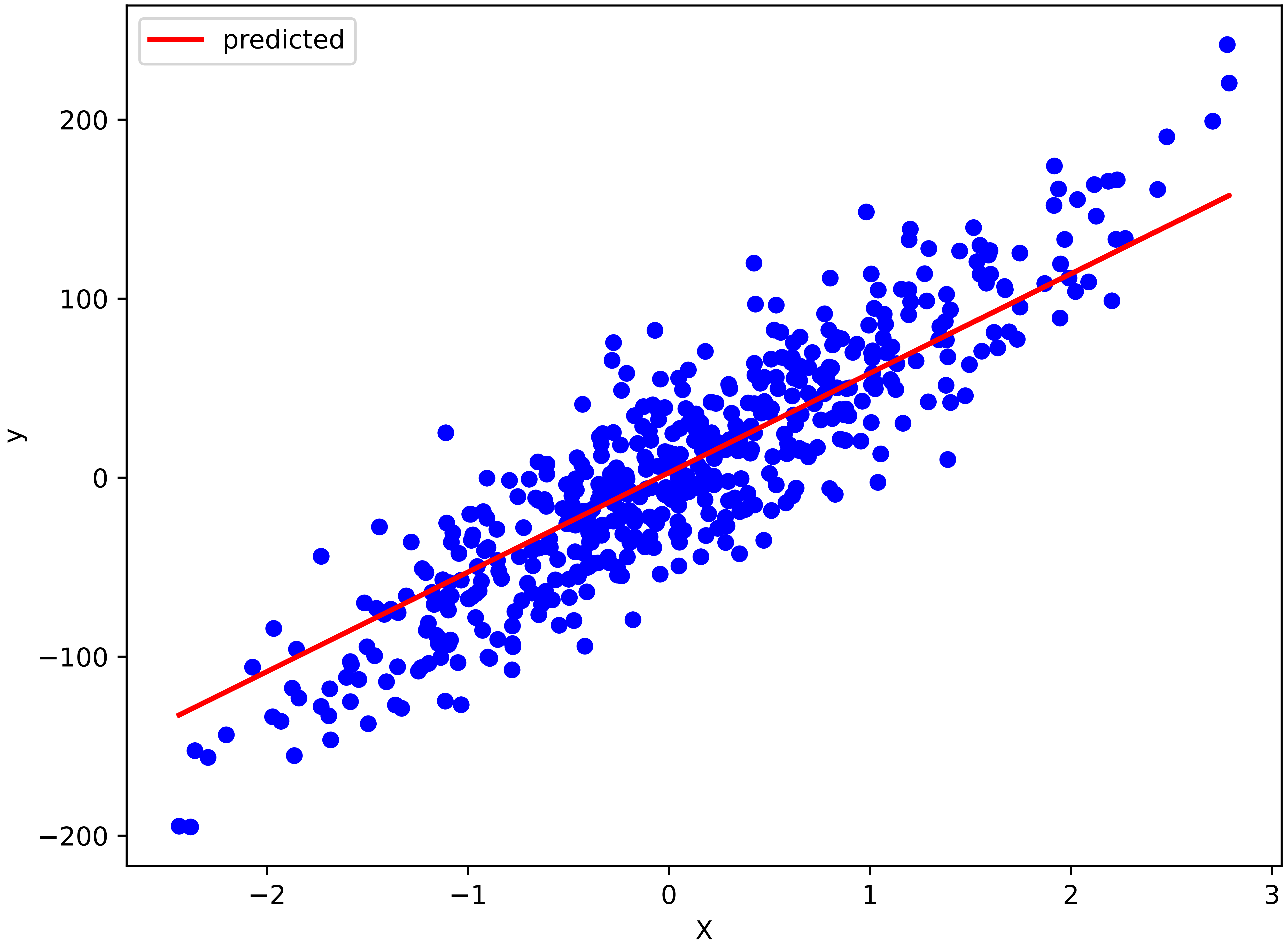
- Some value of weight 'w' and bias 'b'.
- These weights controls the shape of the line.
- MSE = 1520 ↓



Linear Regression

Loss function

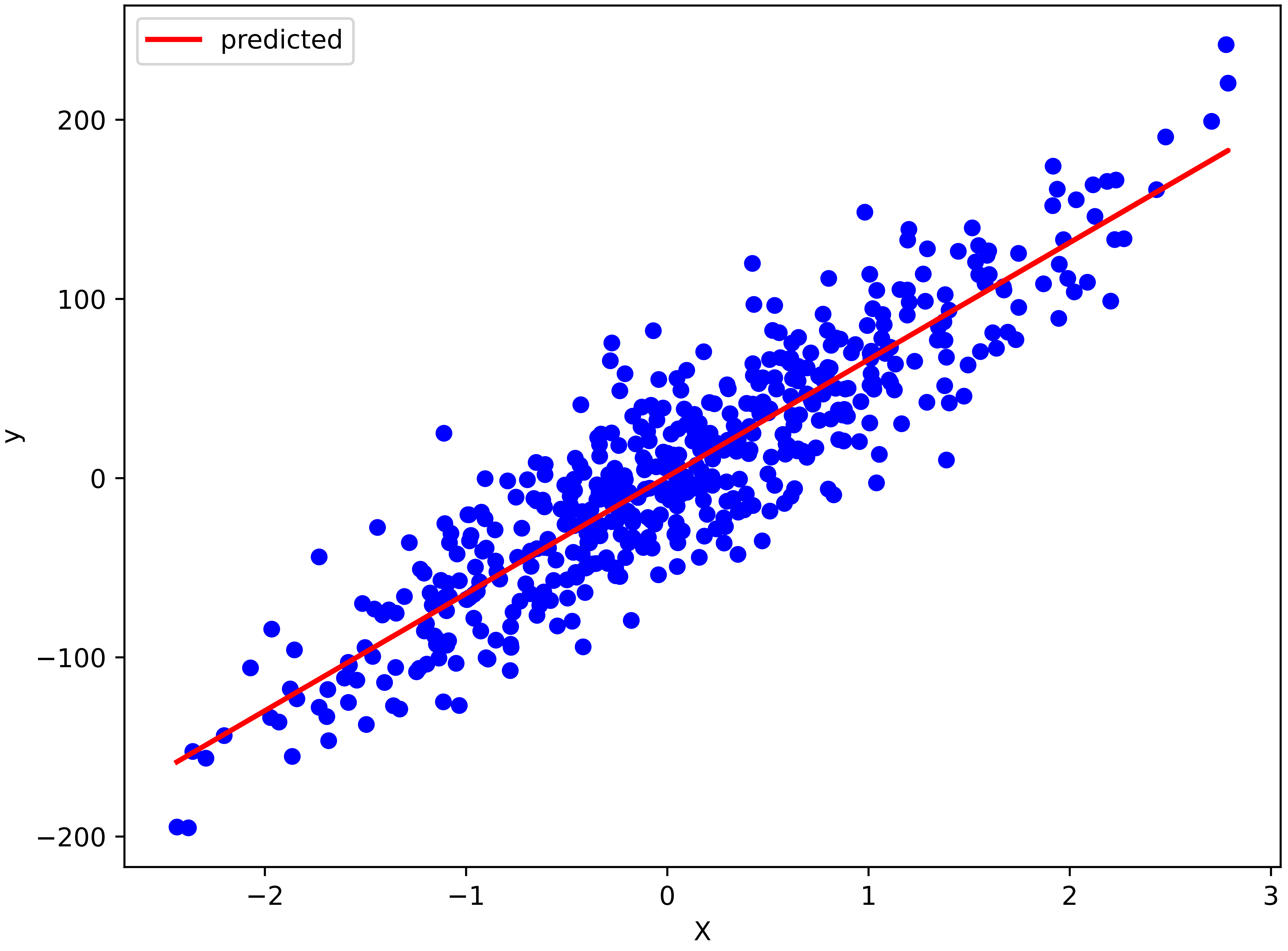
- Some value of weight 'w' and bias 'b'.
- These weights controls the shape of the line.
- MSE = 956 ↓
- Almost !



Linear Regression

Loss function

- ***Optimal*** value of weight 'w' and bias 'b'.
- We can observe, that most fited line is realy ***"close"*** to the whole data points, compared to the first line (in slide 6).
- MSE = 903 ★



Linear Regression

Gradient Descent Algorithm

- In the previous slide, we observed that GDA algorithm progressively converge toward optimal weights, corresponding to the ***minimum mean squared error (MSE)*** value.
- Before introducing this algorithm, let's first discuss the underlying intuition.

Linear Regression

Gradient Descent Algorithm

- In the previous slide, we observed that GDA algorithm progressively converge toward optimal weights, corresponding to the ***minimum mean squared error (MSE)*** value.
- Before introducing this algorithm, let's first discuss the underlying intuition.

- We first define $J(W) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, y'_i) = \frac{1}{n}\sum_{i=1}^{n} [y_i - y'_i]^2$ (i.e. MSE equation)
- In matrix form : $J(W) = \frac{1}{n}\sum_{i=1}^{n} [Y - (X \,.\, W + b)]^2$
- We want to find weights W (and bias b) that ***minimizes*** $J(W)$, how ?
- Randomly … but we can find a better and direct way : ***gradient calculation***

Linear Regression

Gradient Descent Algorithm

- In the previous slide, we observed that GDA algorithm progressively converge toward optimal weights, corresponding to the ***minimum mean squared error (MSE)*** value.
- Before introducing this algorithm, let's first discuss the underlying intuition.

- We first define $J(W) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, y'_i) = \frac{1}{n}\sum_{i=1}^{n} [y_i - y'_i]^2$ (i.e. MSE equation)
- In matrix form : $J(W) = \frac{1}{n}\sum_{i=1}^{n} [Y - (X\,.\,W + b)]^2$
- We want to find weights W (and bias b) that ***minimizes*** $J(W)$, how ?
- Randomly … but we can find a better and direct way : ***gradient calculation***

- The gradient of a function at a given point shows the direction of the fastest increase in the function's value.
- Gradient descent uses this property to find the function's minimum by systematically moving to the ***opposite direction.***

Linear Regression

Gradient Descent Algorithm

- In the previous slide, we observed that GDA algorithm progressively converge toward optimal weights, corresponding to the ***minimum mean squared error (MSE)*** value.
- Before introducing this algorithm, let's first discuss the underlying intuition.

‣ Gradient of ***weights*** :

$$\frac{\delta J(W)}{\delta W} = \frac{\delta}{\delta W}\left[\frac{1}{n}\sum_{i=1}^{n}[y_i - (x_i \,.\, W + b)]^2\right]$$

$$= \frac{2}{n}\sum_{i=1}^{n} - x_i[y_i - (x_i \,.\, W + b)]$$

$$= \frac{2}{n}\sum_{i=1}^{n} - x_i[y_i - y_i']$$

$$= -\frac{2}{n}\, X^T(Y - Y') \quad \text{(matrix form)}$$

‣ Gradient of ***bias*** :

$$\frac{\delta J(W)}{\delta b} = \frac{\delta}{db}\left[\frac{1}{n}\sum_{i=1}^{n}[y_i - (x_i \,.\, W + b)]^2\right]$$

$$= \frac{2}{n}\sum_{i=1}^{n}[y_i - (x_i \,.\, W + b)]$$

$$= \frac{2}{n}\sum_{i=1}^{n} - [y_i - y_i']$$

$$= -\frac{2}{n}\,(Y - Y') \quad \text{(matrix form)}$$

Linear Regression

Gradient Descent Algorithm

➡ Entry : Data input values X of size (**n samples, m features**), data target values Y of size $(n, 1)$
MaxIterations, learning rate α

➡ Output : Weights W of size $(1,\ m)$, and bias b

✤ Training (Learning weights)

(1) **Initialisation :** random values for vector W, and bias b

(2) **While** termination condition :

For each sample $(x_i,\ y_i)$ **:**

- Calculate predicted value $y_i' = x_i\,.\,W + b$
- Compute gradient $\dfrac{\delta J(W)}{\delta W}$ and $\dfrac{\delta J(W)}{\delta b}$
- Updating weights : $W = W - \alpha \dfrac{\delta J(W)}{\delta W}$
- Updating bias : $b = b - \alpha \dfrac{\delta J(W)}{\delta b}$

(3) **Return** W, b

Linear Regression

Gradient Descent Algorithm

➡ Entry : Data input values X of size (*n* **samples,** *m* **features),** data target values Y of size $(n, 1)$
MaxIterations, learning rate α

➡ Output : Weights W of size $(1,\ m)$, and bias b

✤ Training (Learning weights)

(1) **Initialisation :** random values for vector W, and bias b

(2) **While** termination condition :

For each sample $(x_i,\ y_i)$ **:**

- Calculate predicted value $y_i' = x_i\,.\,W + b$
- Compute gradient $\dfrac{\delta J(W)}{\delta W}$ and $\dfrac{\delta J(W)}{\delta b}$
- Updating weights : $W = W - \alpha \dfrac{\delta J(W)}{\delta W}$
- Updating bias : $b = b - \alpha \dfrac{\delta J(W)}{\delta b}$

(3) **Return** W, b

✤ Learning rate $\alpha \in [0,1]$

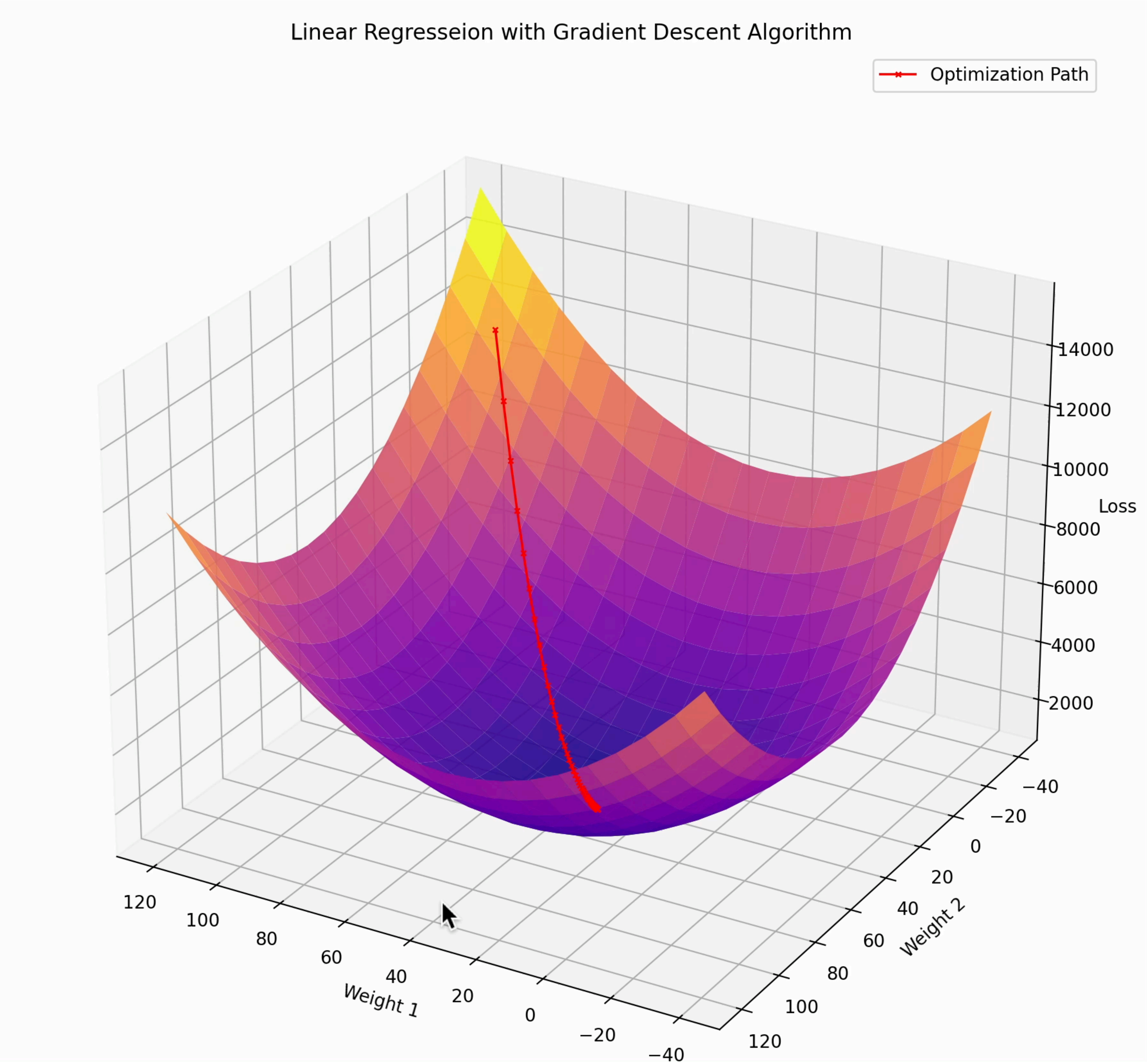
- needs to be adjusted carrefully
- Big value => Big updating steps
- Small value => Small updating steps

★ Prediction (Testing model)

Simply calculate $y_i' = x_i\,.\,W + b$

Linear Regression
GDA & Loss landscape

- *Observe here, a model with two weights (not a single weight as in the former example !).*
- *For each paire (weight 1, weight 2), MSE value (third axis) is calculated. Obtaining MSE landscape.*
- *So the GDA algorithm : starts from some value of weights, then minimize progressively MSE error until arriving to optimal value.*
- *The GDA algorithm trajectory is ploted in red.*
- *MSE is a convex function !*



Practical Activity 1 (part I)

Linear Regression

- Download ***LAB***1 support from Moodle : https://moodle.myefrei.fr/course/view.php?id=14646
- LABs comprise groups of up to 2 students.
- Bonuses can be attributed ! if the teacher considers the results to be relevant.

Logistic Regression

Formalism

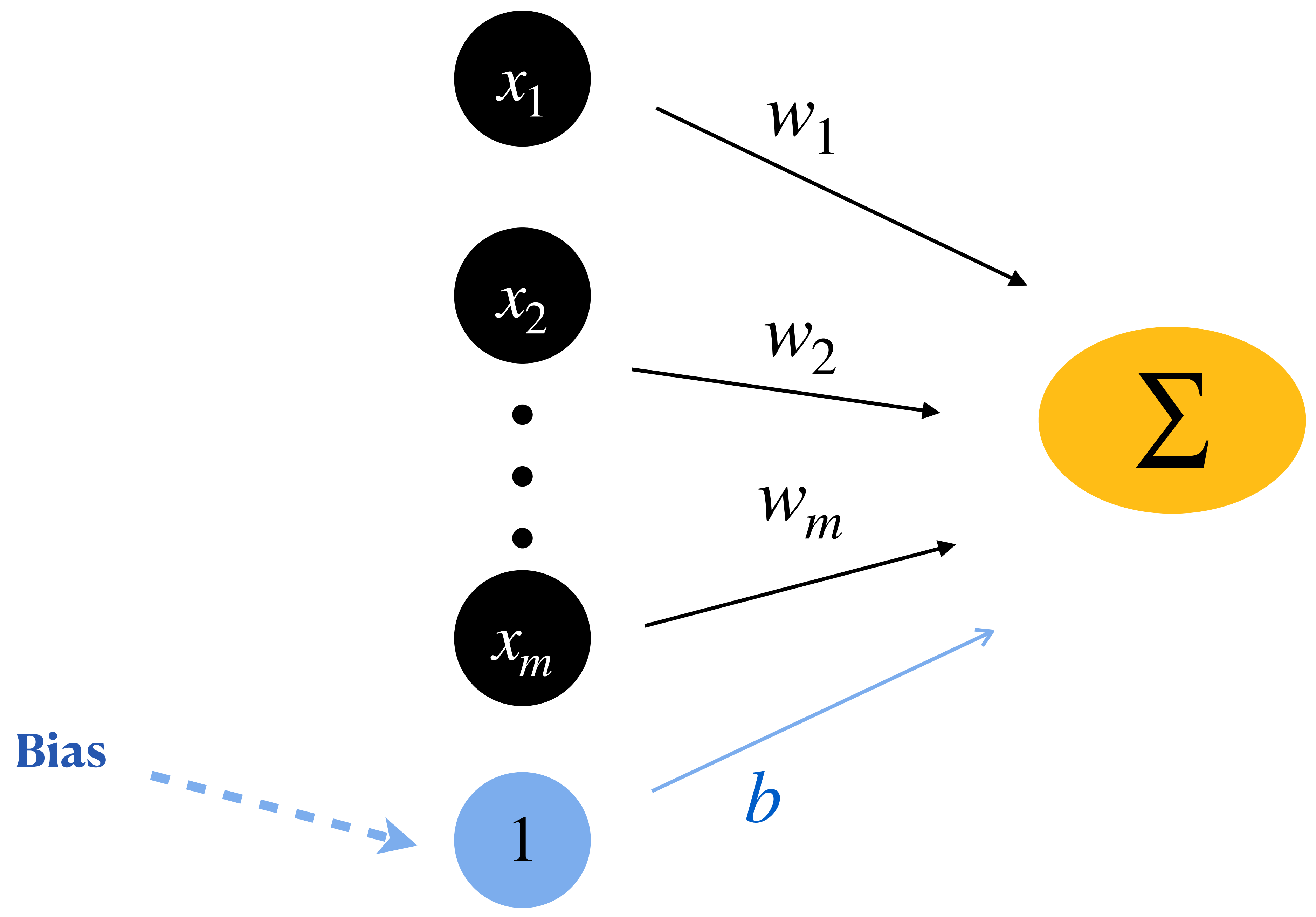
- One of fundamental methods in ***superviserd Machine Learning***.
- Used for ***binary classification tasks !***
- Outcome variable is categorical with two possible outcomes (e.g., 0 or 1)
- It models the probability that an observation belongs to one of the categories.

Logistic Regression

Formalism

- One of fundamental methods in ***superviserd Machine Learning***.
- Used for ***binary classification tasks !***
- Outcome variable is categorical with two possible outcomes (e.g., 0 or 1)
- It models the probability that an observation belongs to one of the categories.

Inputs Weights Output



$$y' = f\left(\sum_{i=1}^{m} x_i w_i + b\right)$$

Where $f(z) = \dfrac{1}{1 + e^{-z}}$

$$y' = f\left(X \,.\, W + b\right)$$

(Matrix version)

Logistic Regression
Formalism

- One of fundamental methods in ***superviserd Machine Learning***.
- Used for ***binary classification tasks !***
- Outcome variable is categorical with two possible outcomes (e.g., 0 or 1)
- It models the probability that an observation belongs to one of the categories.

★ General formulation, *the inference function* : $y = f(X \,.\, W + b)$ where $f(z) = \dfrac{1}{1 + e^{-z}}$

- y is the dependent variable : continous values we want to predict
- X represent a data set of n samples and m features (independent variables).
- W is a vector of weights ***(the unknown, we want to learn)***, of size $(1,m)$.
- b is a scalar value called the bias ***(unknown value, we want to learn)***.

★ The goal is to find vector of weights W and scalar term b

Logistic Regression

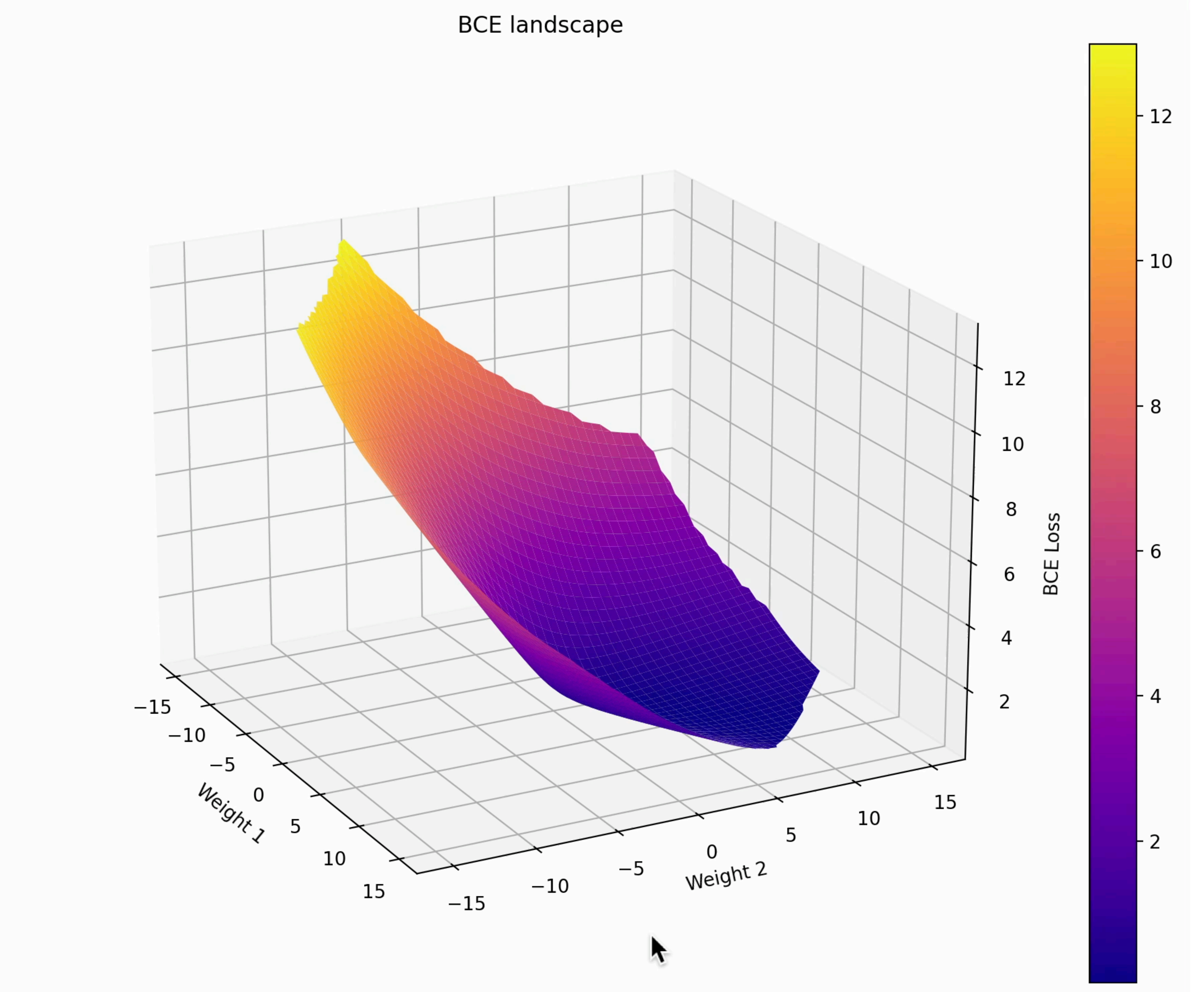
Loss function

- Logistic regression does not use MSE because its output is probabilistic bounded between 0 and 1.
- It uses a loss function known as the log loss (or ***Binary Cross-Entropy loss***) for binary classification.
- It measures the penalty for a given class prediction in terms of the distance from the actual label.

➡ The formula :

$$J(W) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i log(y_i') + (1-y_i) log(1-y_i')\right]$$

➡ ***BCE*** is a convex function, for simple LR case !



Logistic Regression

Gradient Descent Algorithm

- As for *MSE* error, *BCE* is also convex which ensures that Gradient Descent Algorithm has a global minimum
- Before introducing this algorithm, let's first discuss the underlying intuition.

- We first define : $J(W) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i log(y_i') + (1-y_i)log(1-y_i')\right]$ (i.e. BCE equation)
- Where : $y_i' = \frac{1}{1+e^{-(X.W+b)}}$ (i.e. the sigmoid function)
- We want to find weights W (and bias b) that ***minimizes*** $J(W)$, we calculat gradients :
- Gradient of weights : $\frac{\delta J(W)}{\delta W} = -\frac{1}{n}X^T(Y-Y')$
- Gradient of bias : $\frac{\delta J(W)}{\delta W} = -\frac{1}{n}(Y-Y')$

- The gradient of a function at a given point shows the direction of the fastest increase in the function's value.
- Gradient descent uses this property to find the function's minimum by systematically moving to the ***opposite direction.***

Logistic Regression
Gradient Descent Algorithm

➡ Entry : Data input values X of size (n **samples,** m **features),** data target values Y of size $(n, 1)$
MaxIterations, learning rate α, ***Threshold*** $\in [0,1]$

➡ Output : Weights W of size $(1,\ m)$, and bias b

✤ Training (Learning weights)

(1) **Initialisation :** random values for vector W, and bias b

(2) **While** termination condition :

For each random sample $(x_i,\ y_i)$ **:**

- Calculate predicted value $y_i' = \left[1 + e^{-(x_i.W+b)}\right]^{-1}$
- Compute gradient $\dfrac{\delta J(W)}{\delta W}$ and $\dfrac{\delta J(W)}{\delta b}$
- Updating weights : $W = W - \alpha \dfrac{\delta J(W)}{\delta W}$
- Updating bias : $b = b - \alpha \dfrac{\delta J(W)}{\delta b}$

(3) **Return** W, b

Logistic Regression

Gradient Descent Algorithm

➡ Entry : Data input values X of size (**n samples, m features**), data target values Y of size $(n, 1)$
MaxIterations, learning rate α, ***Threshold*** $\in [0,1]$

➡ Output : Weights W of size $(1,\ m)$, and bias b

✤ Training (Learning weights)

(1) **Initialisation :** random values for vector W, and bias b

(2) **While** termination condition :

For each random sample $(x_i,\ y_i)$ **:**

- Calculate predicted value $y_i' = \left[1 + e^{-(x_i.W+b)}\right]^{-1}$
- Compute gradient $\dfrac{\delta J(W)}{\delta W}$ and $\dfrac{\delta J(W)}{\delta b}$
- Updating weights : $W = W - \alpha \dfrac{\delta J(W)}{\delta W}$
- Updating bias : $b = b - \alpha \dfrac{\delta J(W)}{\delta b}$

(3) **Return** W, b

✤Learning rate $\alpha \in [0,1]$

- needs to be adjusted carrefully
- Big value => Big updating steps
- Small value => Small updting steps

★Prediction (Testing model)

- Simply calculate $y_i' = \left[1 + e^{-(x_i.W+b)}\right]^{-1} \in [0,1]$
- **if** $(y_i' \geq Threshold)$ **then** x_i **is a cat**
  else x_i **is a dog**

★Accuracy

An accuracy function may be used to verify the accuracy of the model : ***predicted classes Vs target classes for each*** x_i.

Summary : Distinctions

	Linear Regression	Logistic Regression
The purpose	**Prediction** of continuous values	Binary **Classification**
The model form	$X.W + b$	$\left[1 + e^{-(X.W+b)}\right]^{-1}$
The loss function	Mean Squared Error (MSE)	Binary Cross-Entropy (BCE)
The algorithm	GDA	GDA
Limitation	Data landscape must be linear	Linear separation of data

For more complicated data landscapes => Polynomial Regression or use of Neural Networks.

Summary : Use cases

Linear Regression	Logistic Regression
Prediction of ***adult characteristics*** based on parent's characteristics	***Fraud detection:*** can help teams identify anomalies in data that may predict fraud
Predicting ***product sales*** volume as a function of price, time of year and store location	***Disease prediction***: Predicting whether a person will develop heart disease based on IMC, smoking habits and genetic predisposition
Predict the ***price of an airline ticket*** as a function of origin, destination, time of year and airline company	***Attrition prediction:*** it could be useful for HR and management to know whether high-performing employees are at risk of leaving the organization

Practical Activity 1 (part II)

Logistic Regression

- Download ***LAB***1 support from Moodle : https://moodle.myefrei.fr/course/view.php?id=14646
- LABs comprise groups of up to 2 students.
- Bonuses can be attributed ! if the teacher considers the results to be relevant.