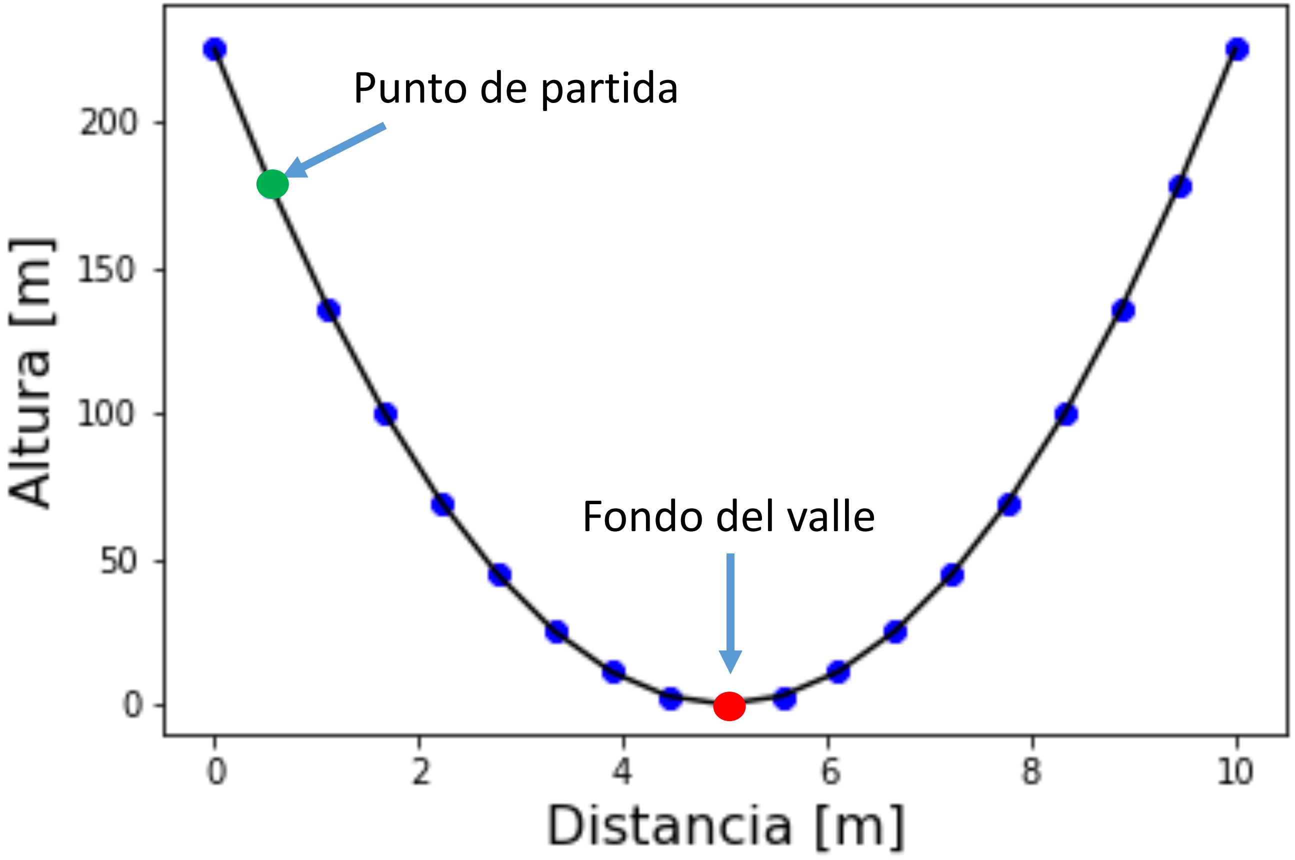
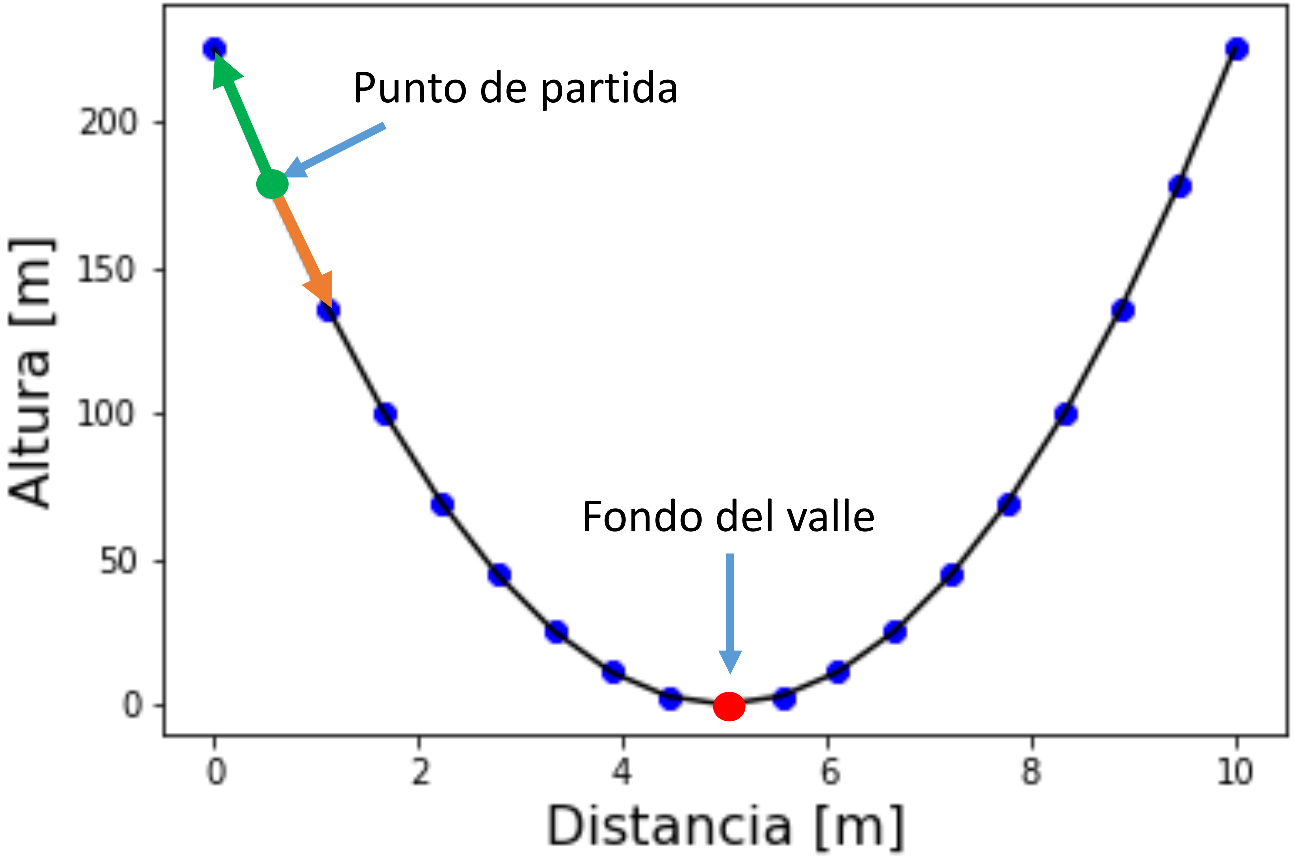
# Tema 3: Descenso de la pendiente

- Descenso de la pendiente.
- Función coste o de perdidas.
- Aplicación del descenso de la pendiente a un perceptrón sigmoide.
- Algoritmo de actualización de pesos en un perceptrón sigmoide.

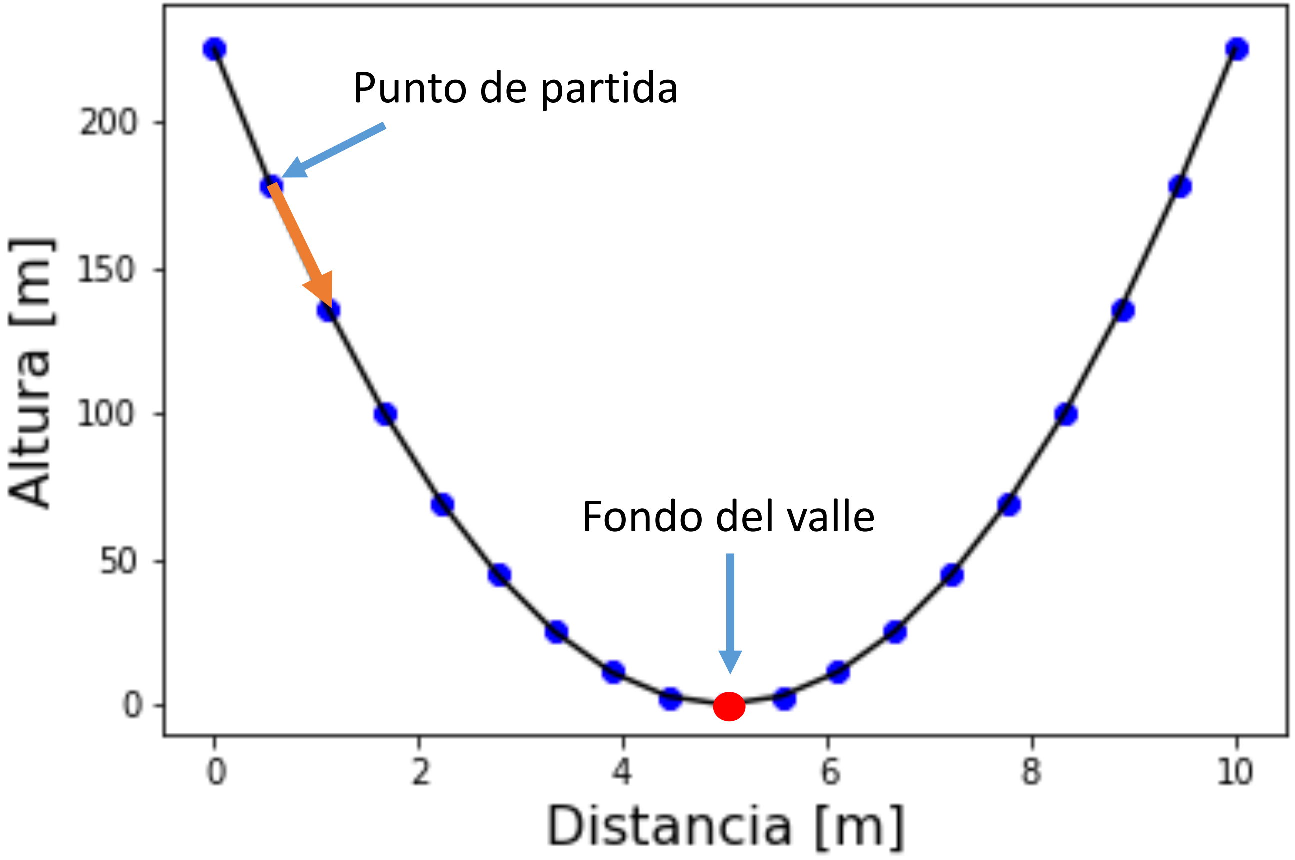
# Descenso de la pendiente:



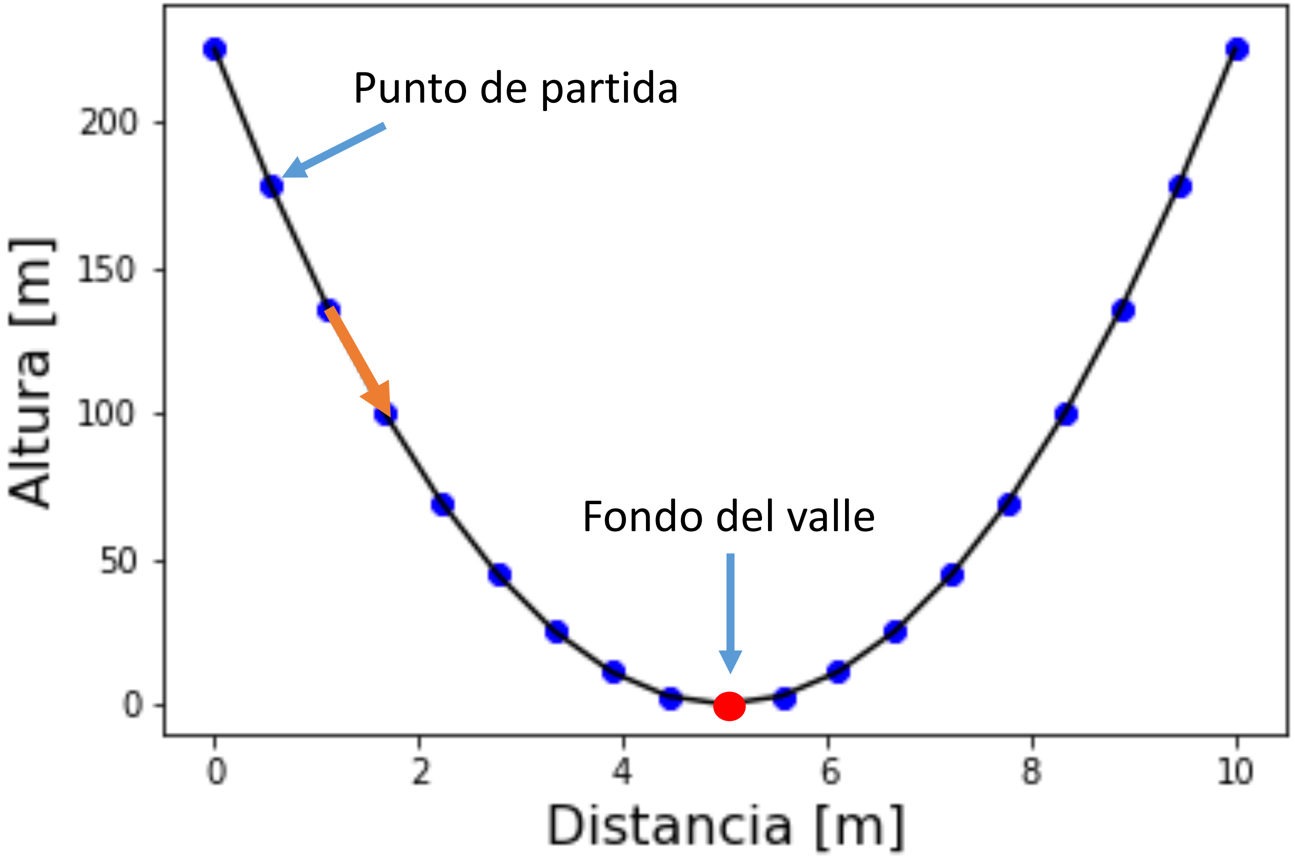
# Descenso de la pendiente:



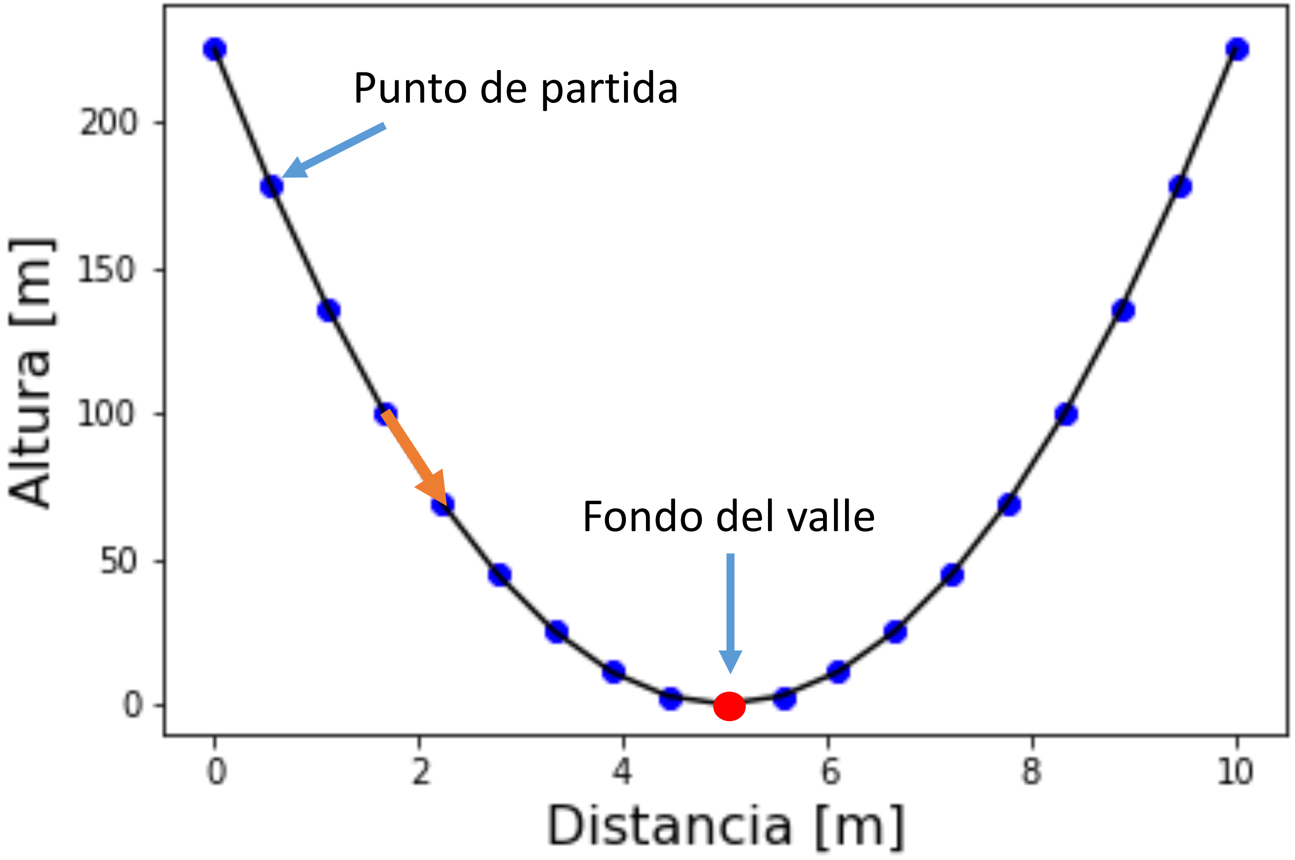
# Descenso de la pendiente:



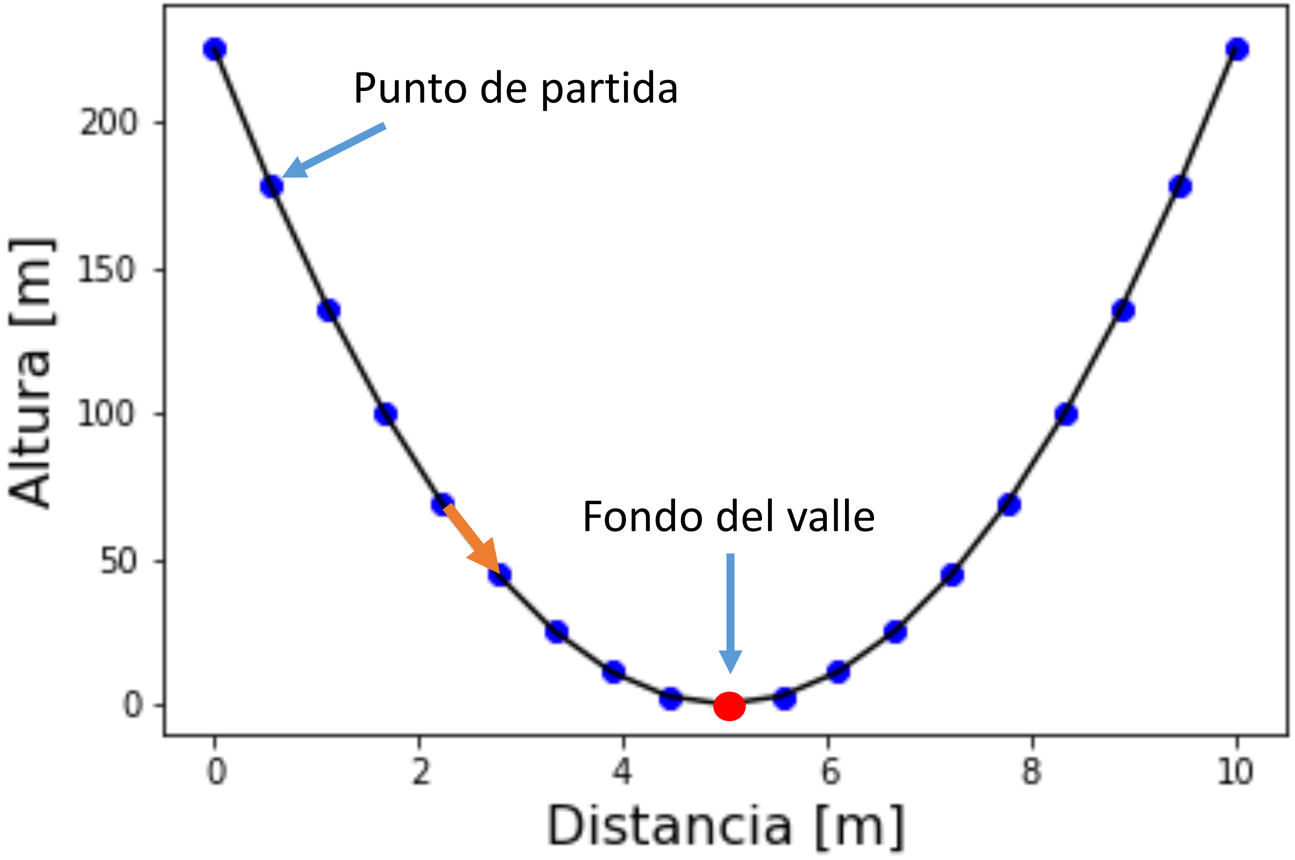
# Descenso de la pendiente:



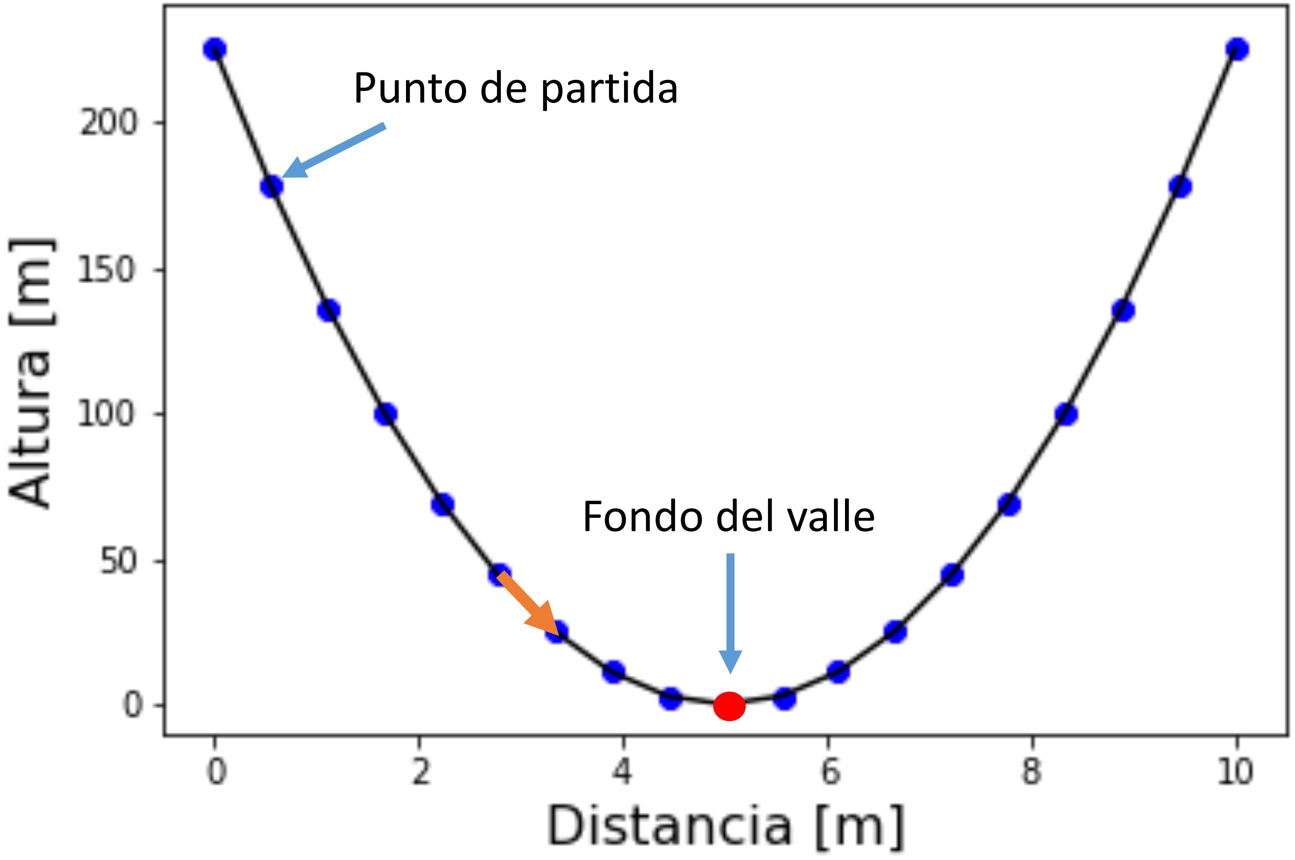
# Descenso de la pendiente:



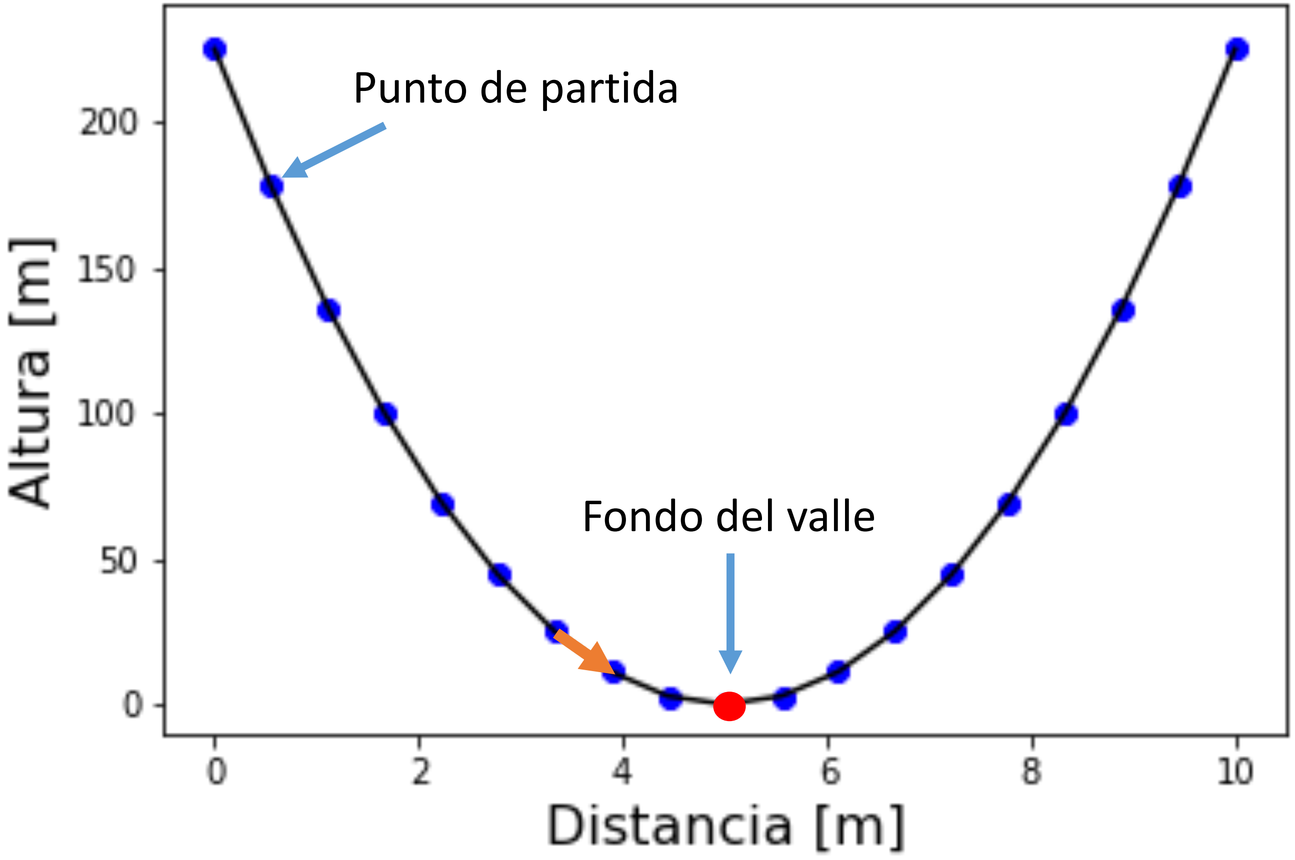
# Descenso de la pendiente:



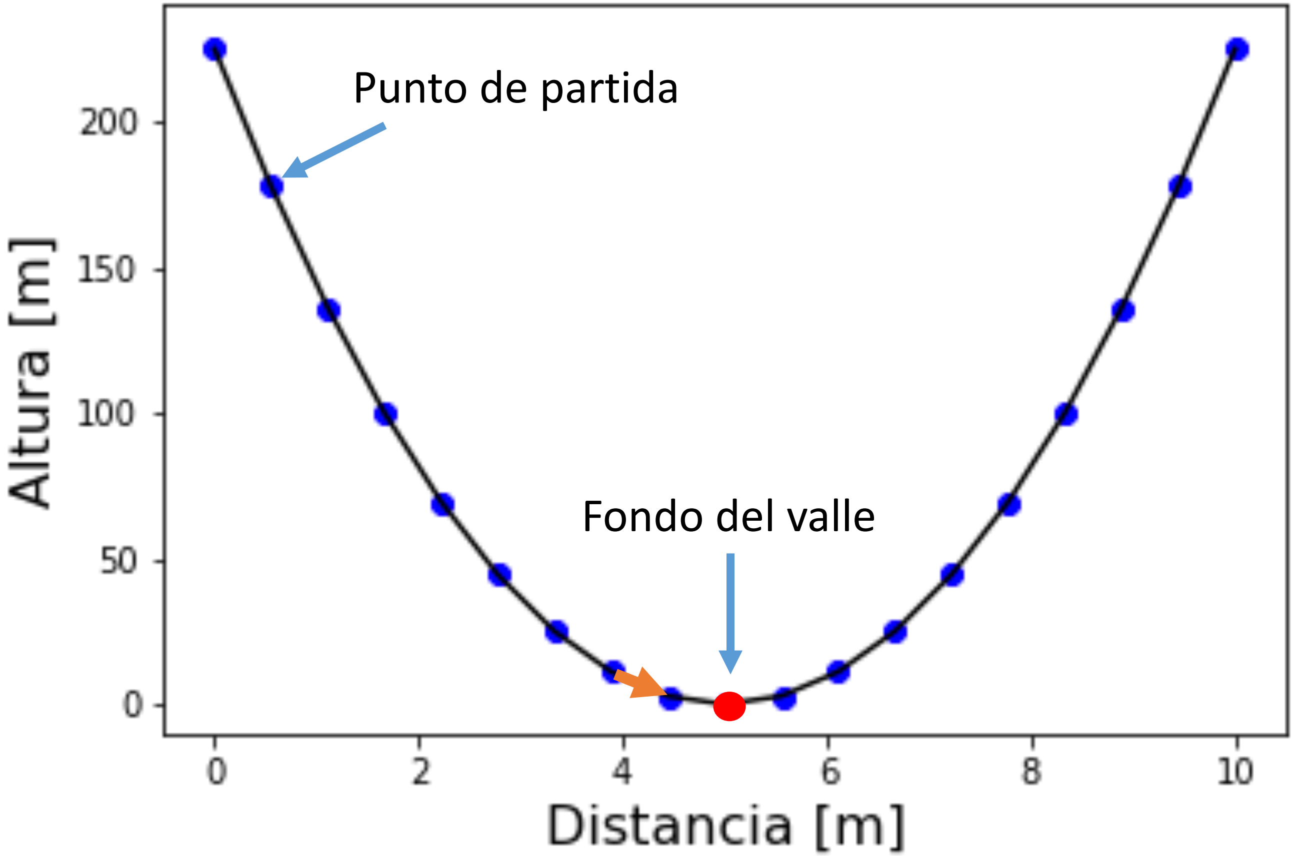
# Descenso de la pendiente:



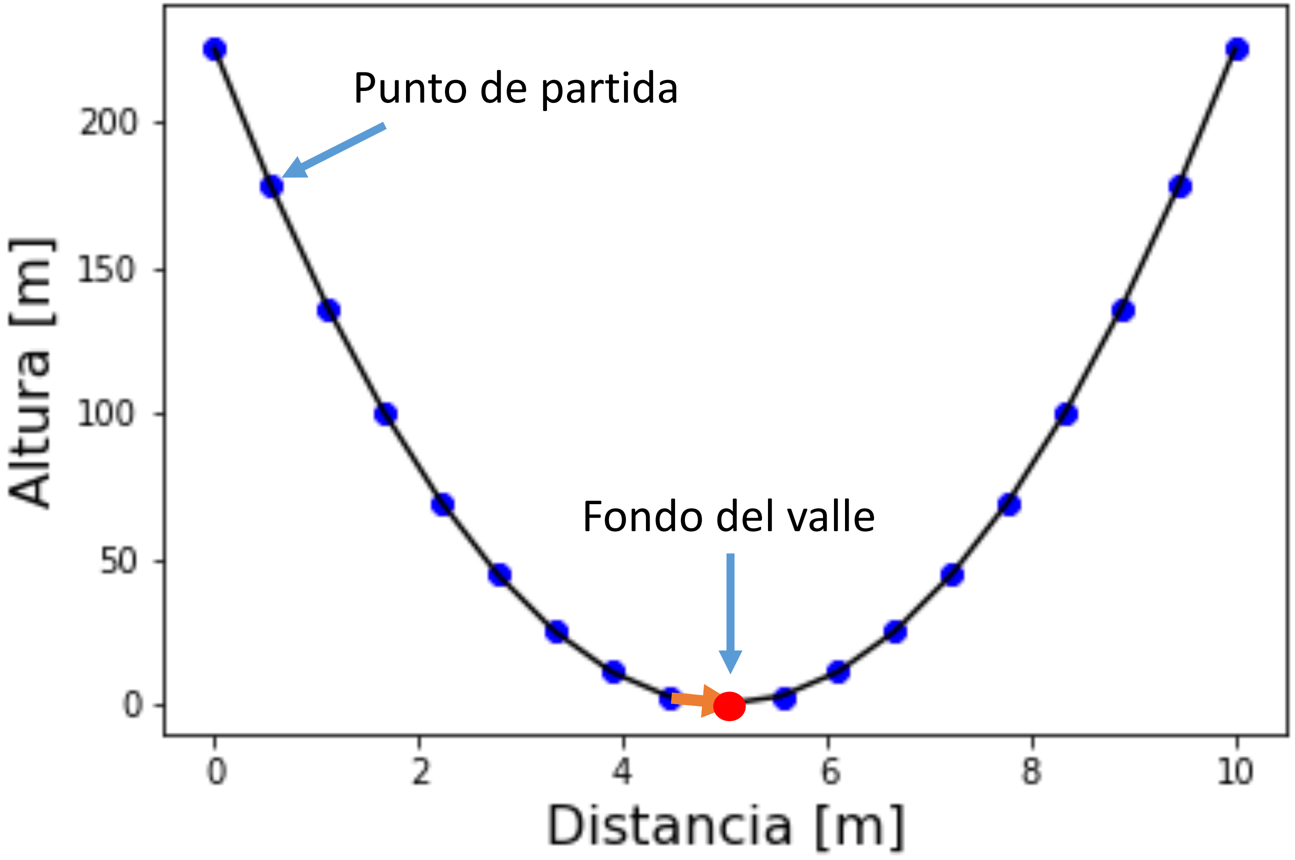
# Descenso de la pendiente:



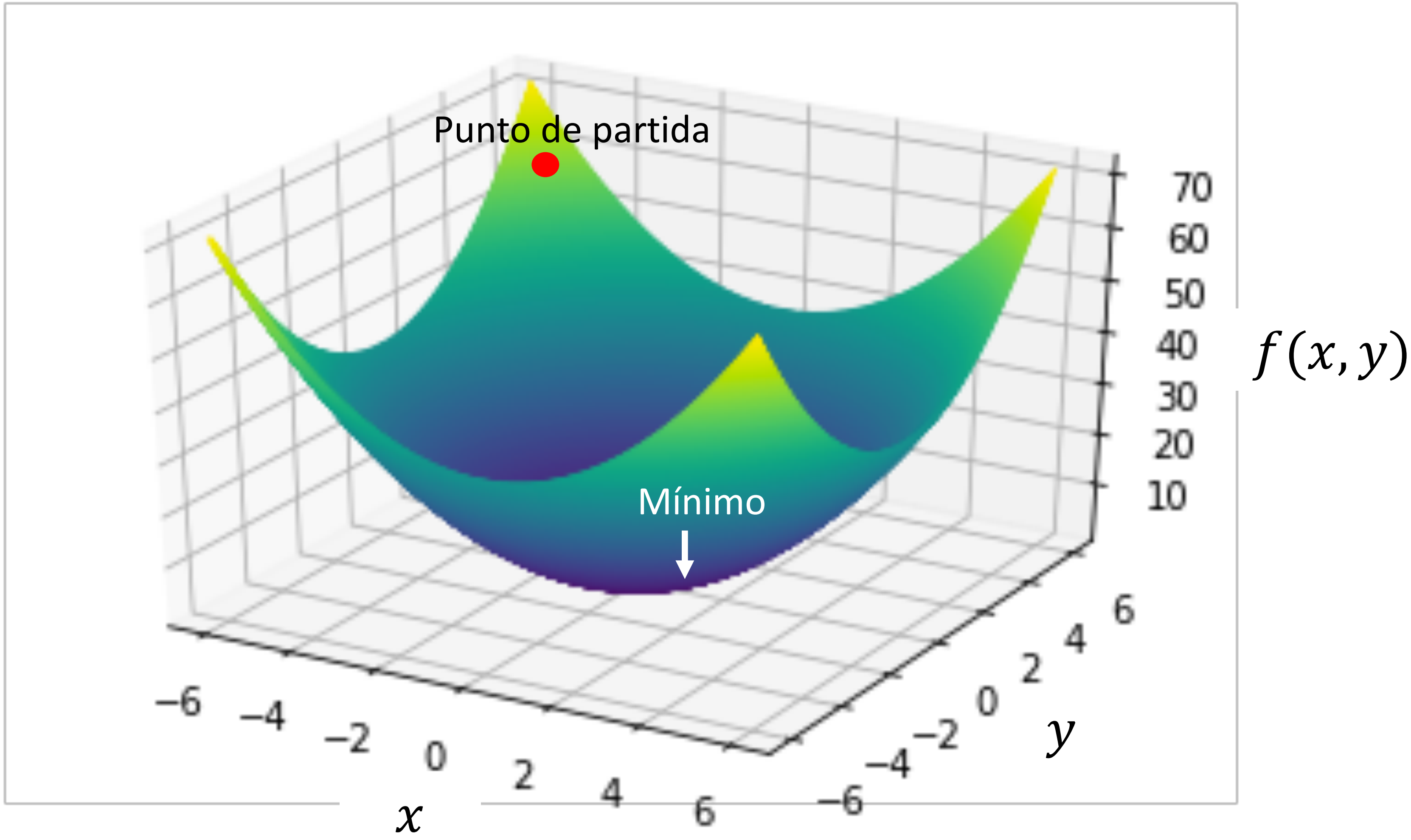
# Descenso de la pendiente:



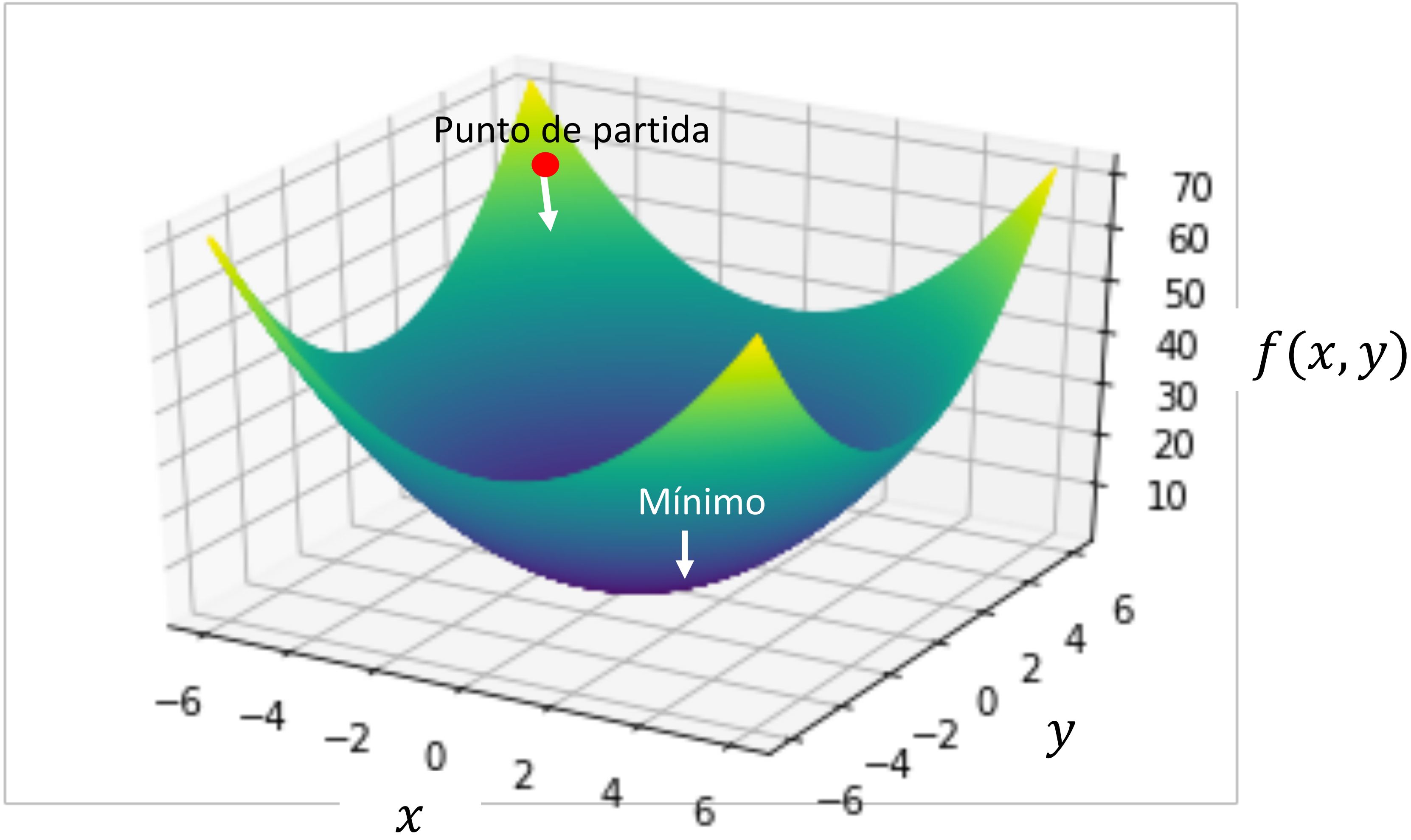
# Descenso de la pendiente:



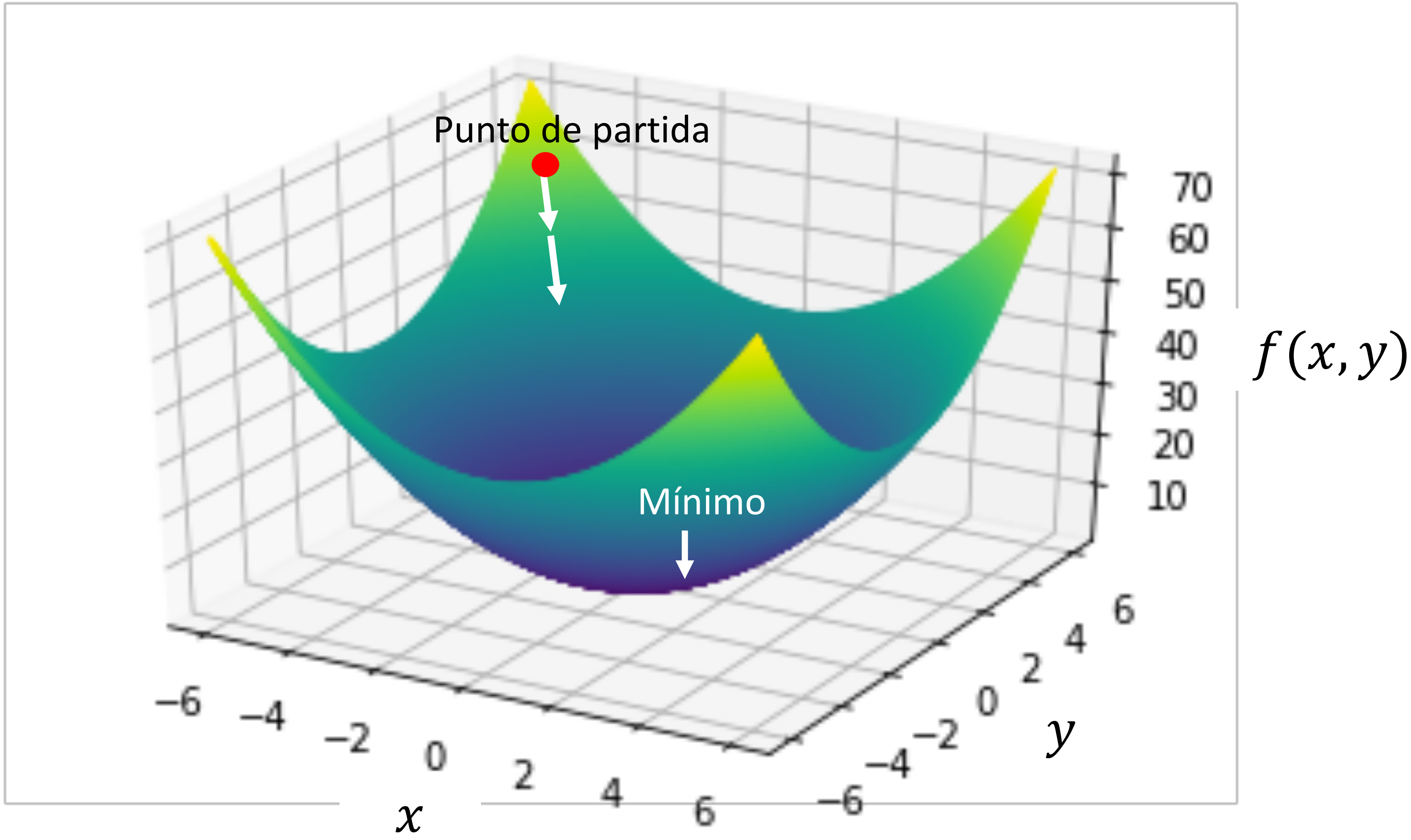
# Descenso de la pendiente:



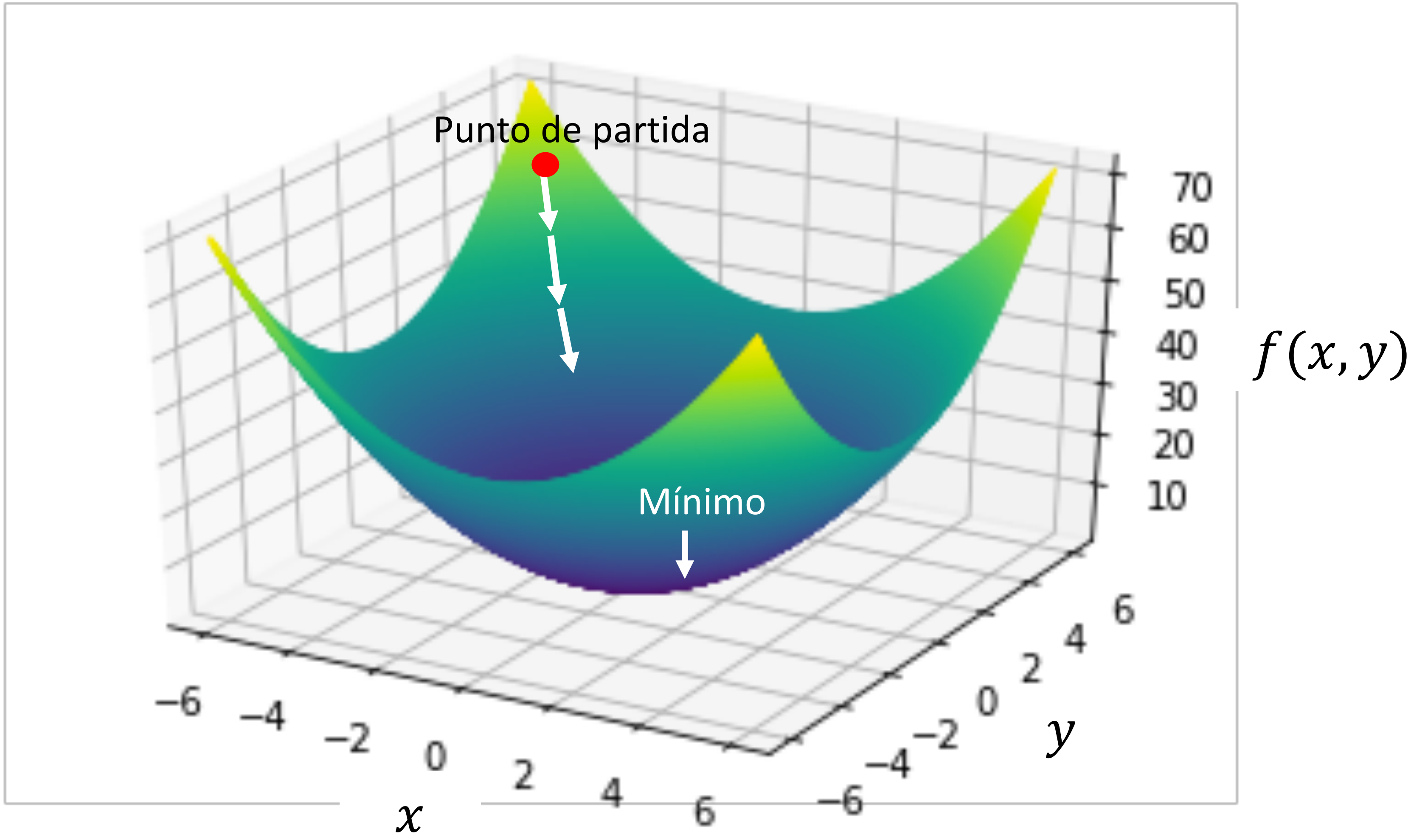
# Descenso de la pendiente:



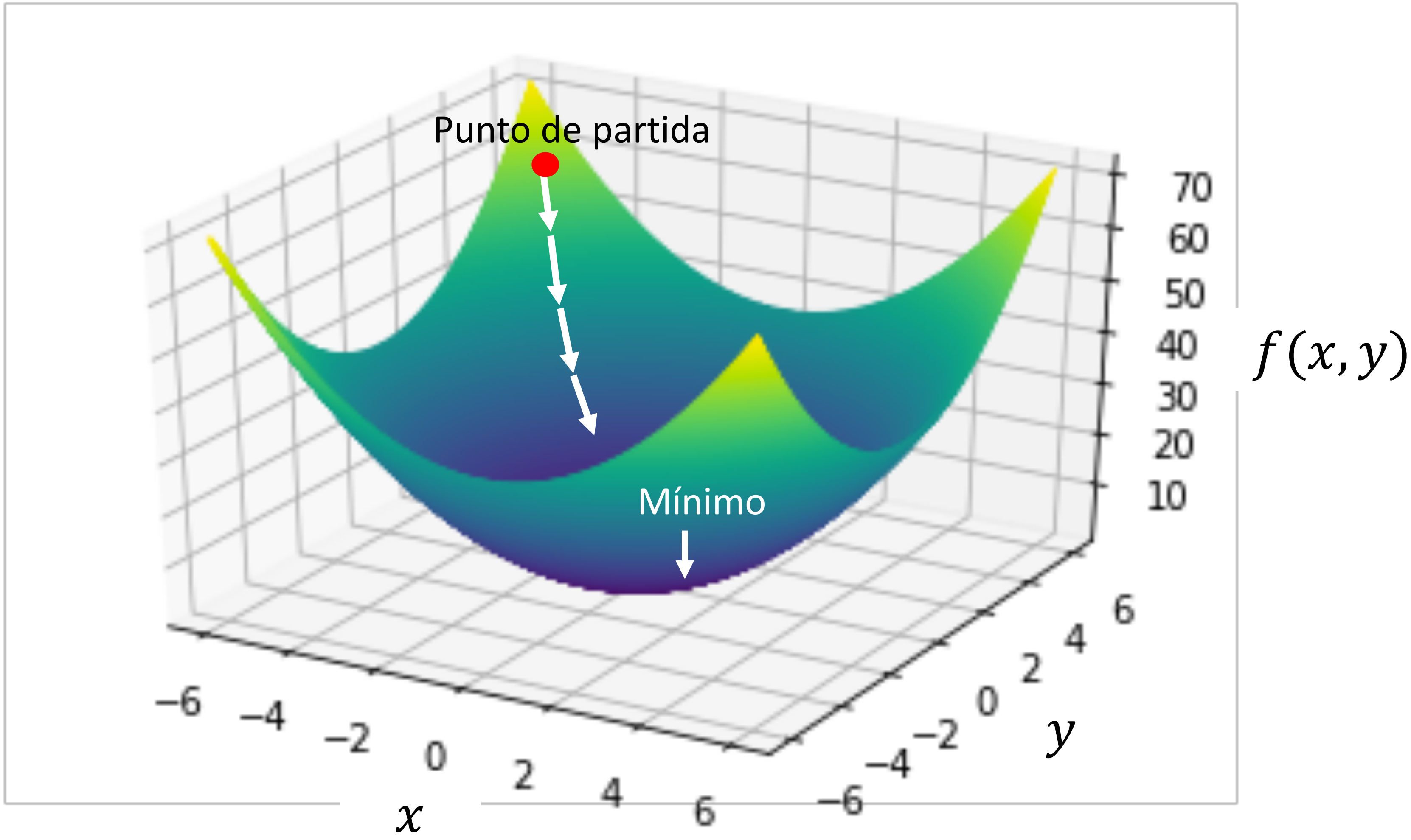
# Descenso de la pendiente:



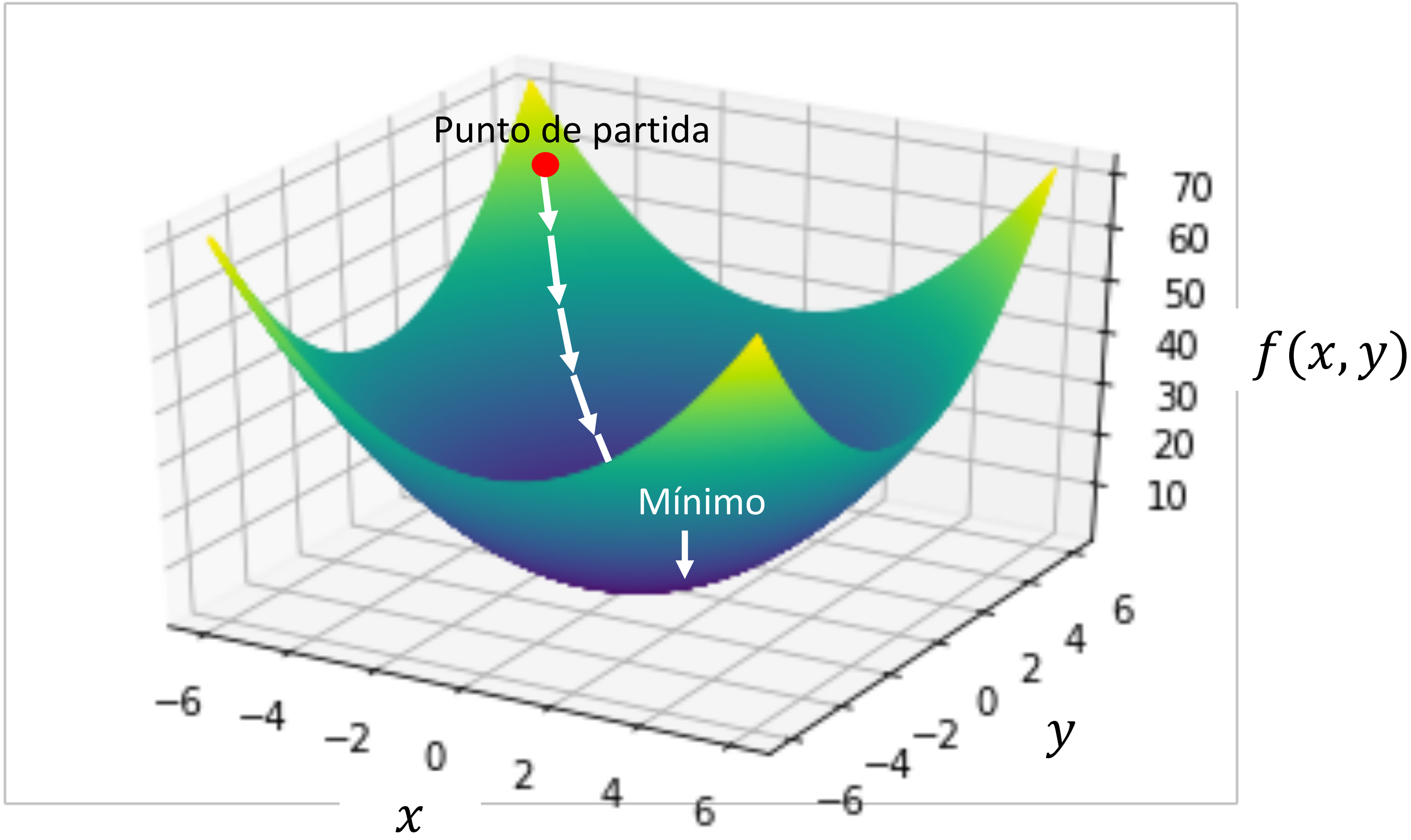
# Descenso de la pendiente:



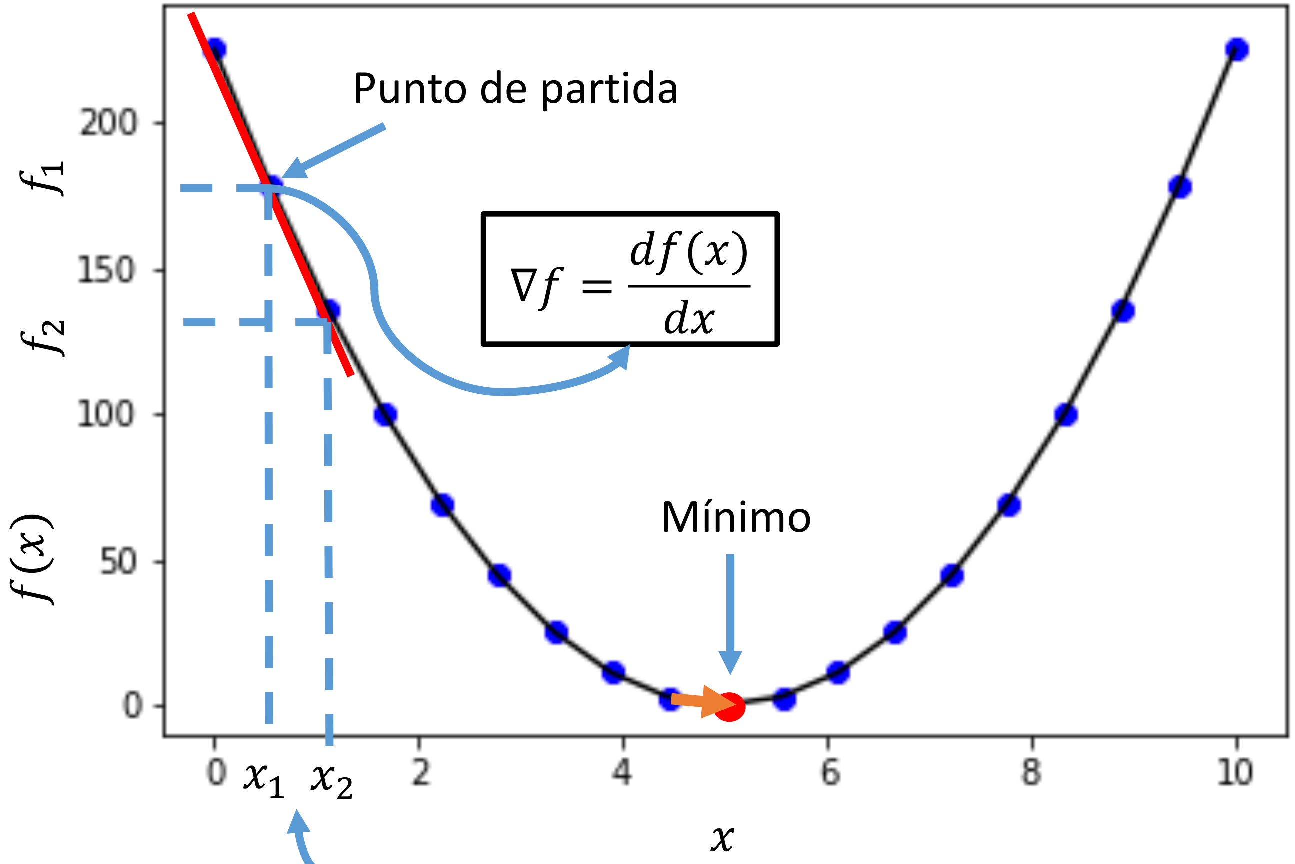
# Descenso de la pendiente:



# Descenso de la pendiente:

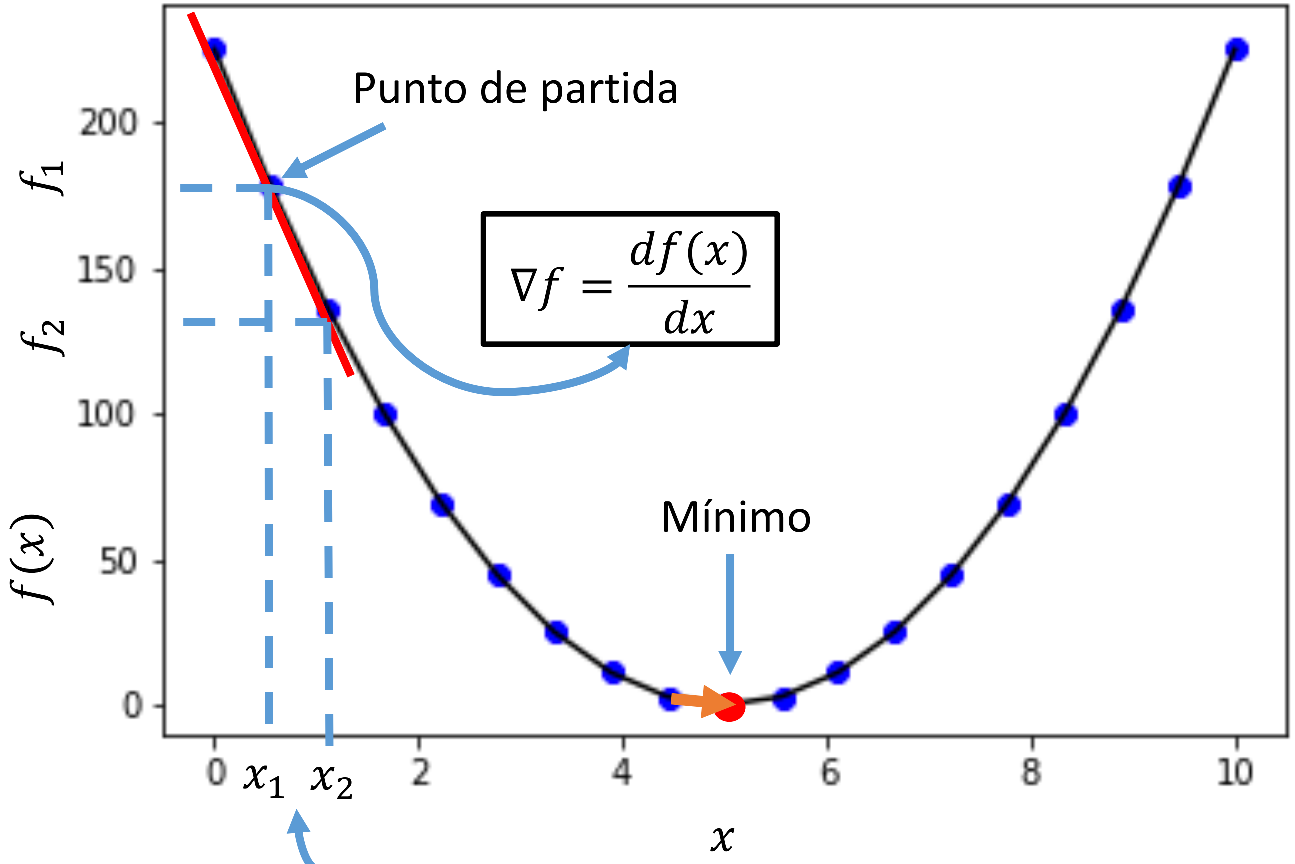


# Descenso de la pendiente:



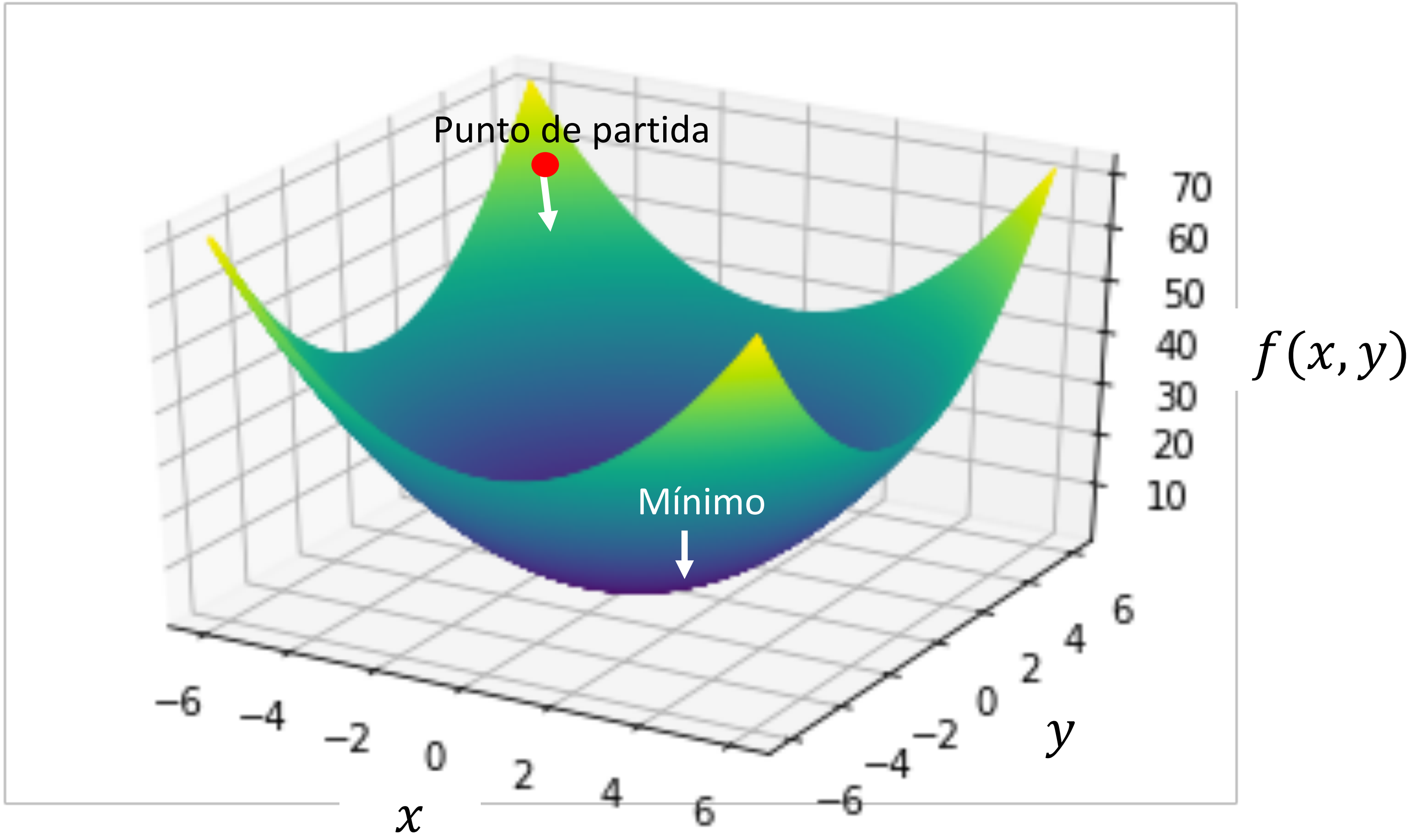

$$x_2 = x_1 + \alpha \frac{df(x)}{dx}$$

# Descenso de la pendiente:



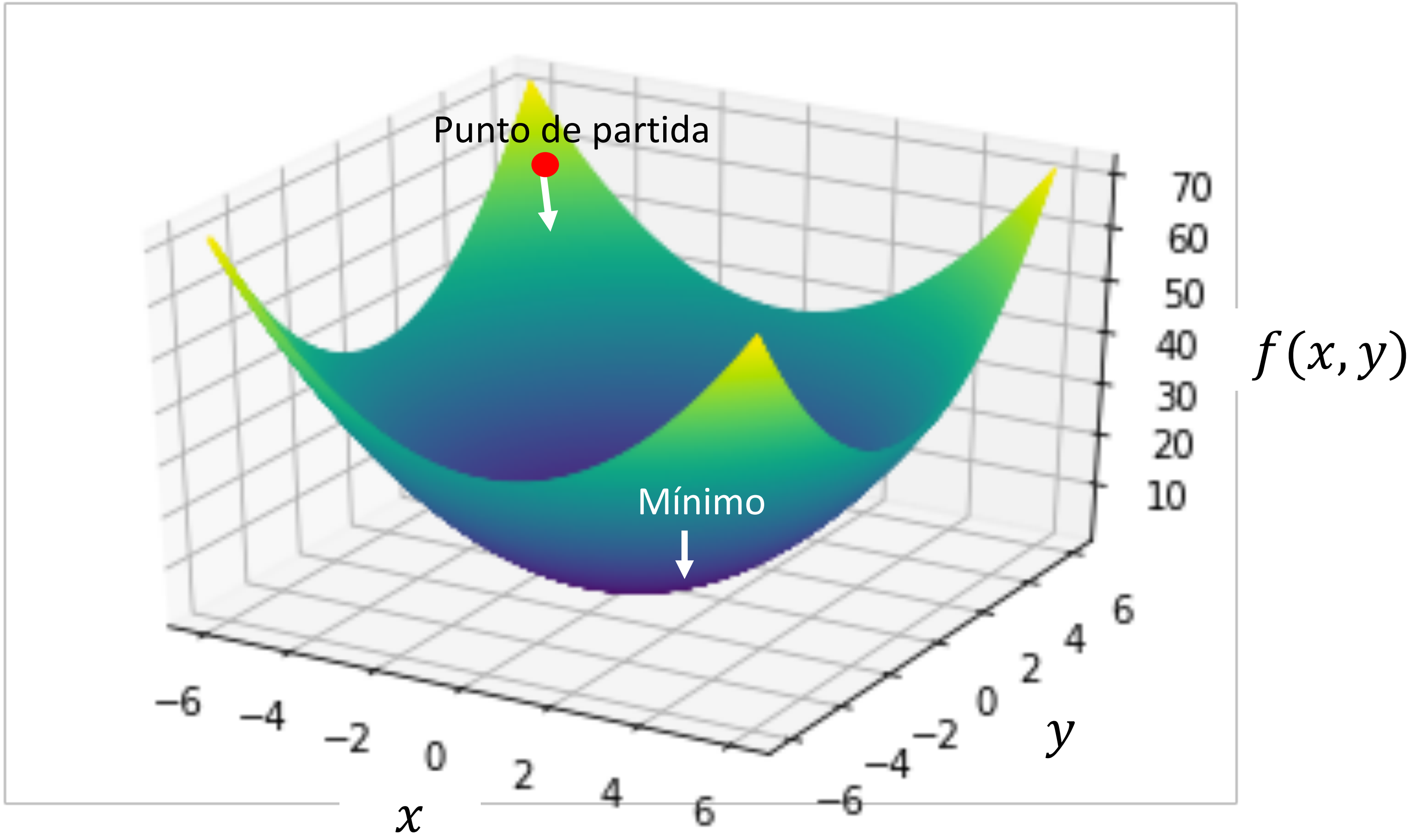

$$x_2 = x_1 + \alpha \frac{df(x)}{dx} \longrightarrow x_{j+1} = x_j + \alpha \frac{df(x)}{dx}$$

# Descenso de la pendiente:




$$\nabla f(\boldsymbol{x},\boldsymbol{y}) = \begin{bmatrix} \dfrac{\partial f(x,y)}{\partial x} \\ \dfrac{\partial f(x,y)}{\partial y} \end{bmatrix}$$

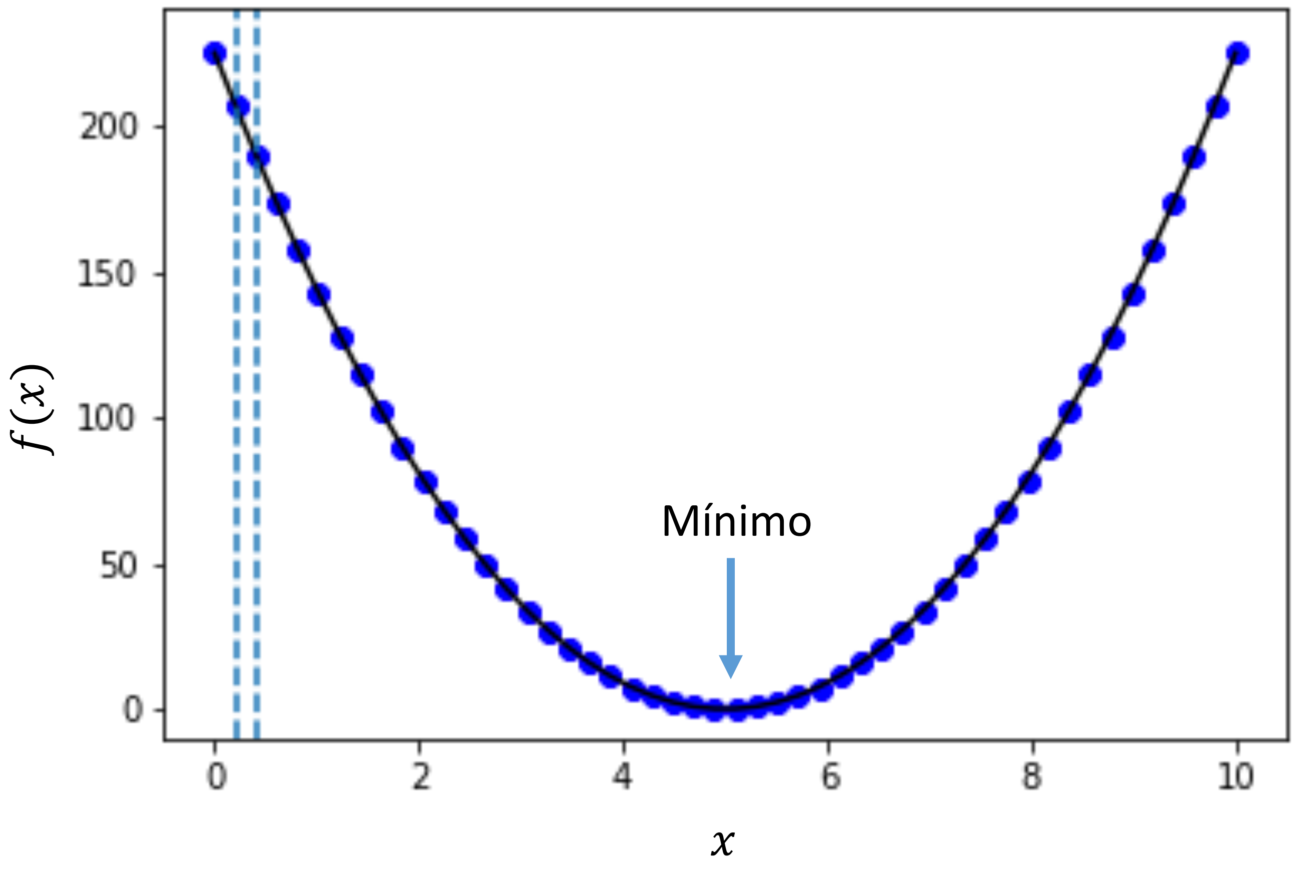
# Descenso de la pendiente:




$$\nabla f(\boldsymbol{x}, \boldsymbol{y}) = \begin{bmatrix} \dfrac{\partial f(x,y)}{\partial x} \\ \dfrac{\partial f(x,y)}{\partial y} \end{bmatrix}$$

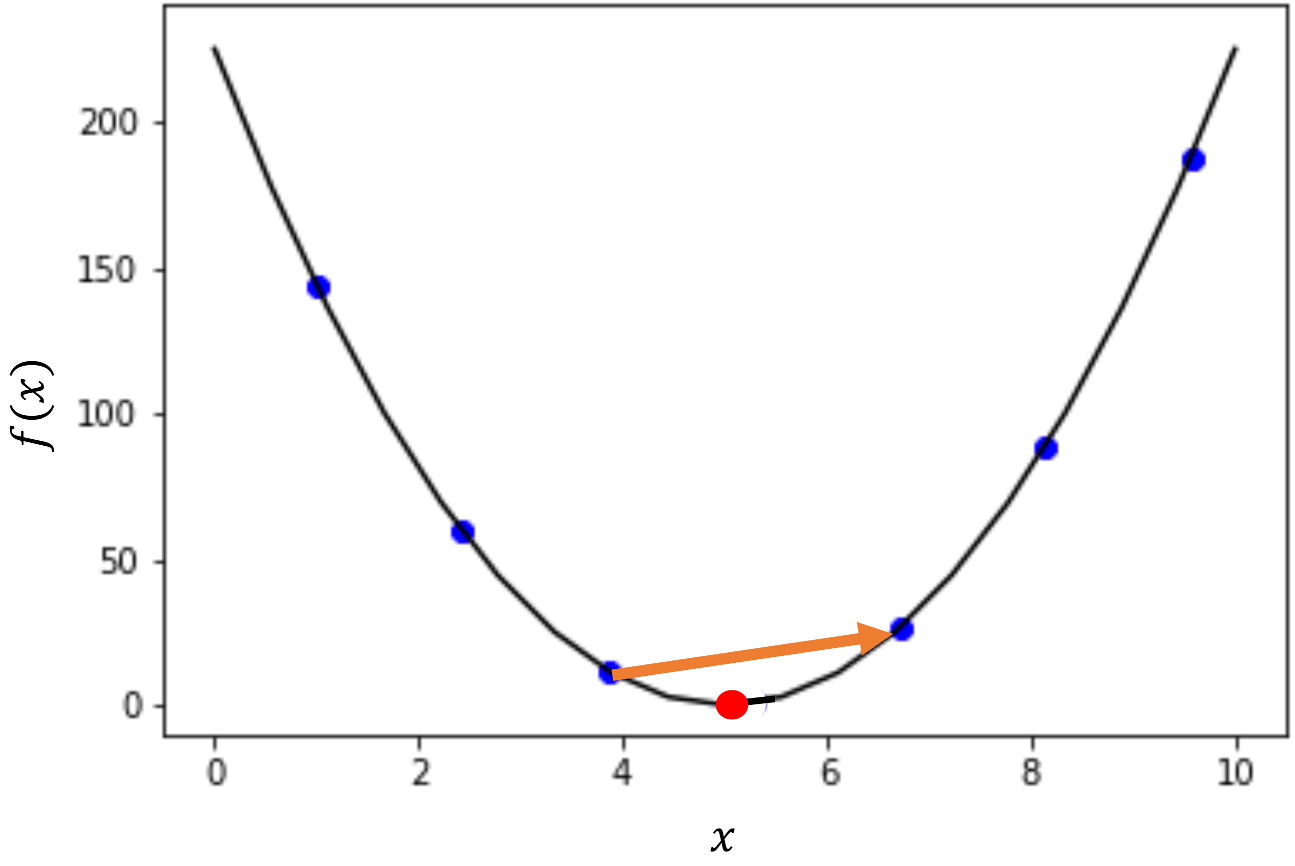
$$f(x_{j+1}, y_{j+1}) \begin{cases} x_{j+1} = x_j + \alpha \dfrac{\partial f(x,y)}{\partial x} \\ \\ y_{j+1} = y_j + \alpha \dfrac{\partial f(x,y)}{\partial y} \end{cases}$$

# Descenso de la pendiente:



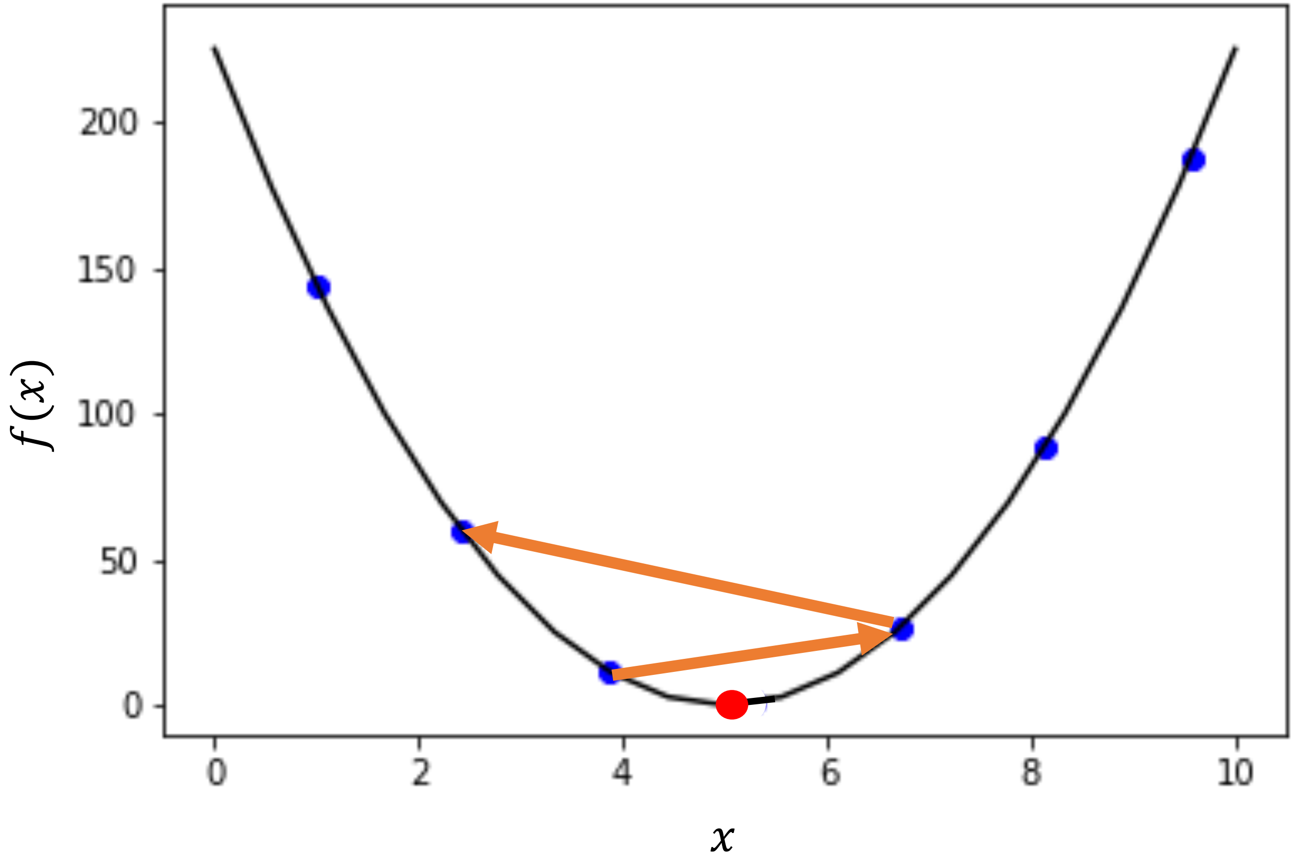

- $\alpha$ pequeños: tiempos de convergencia lentos.

# Descenso de la pendiente:



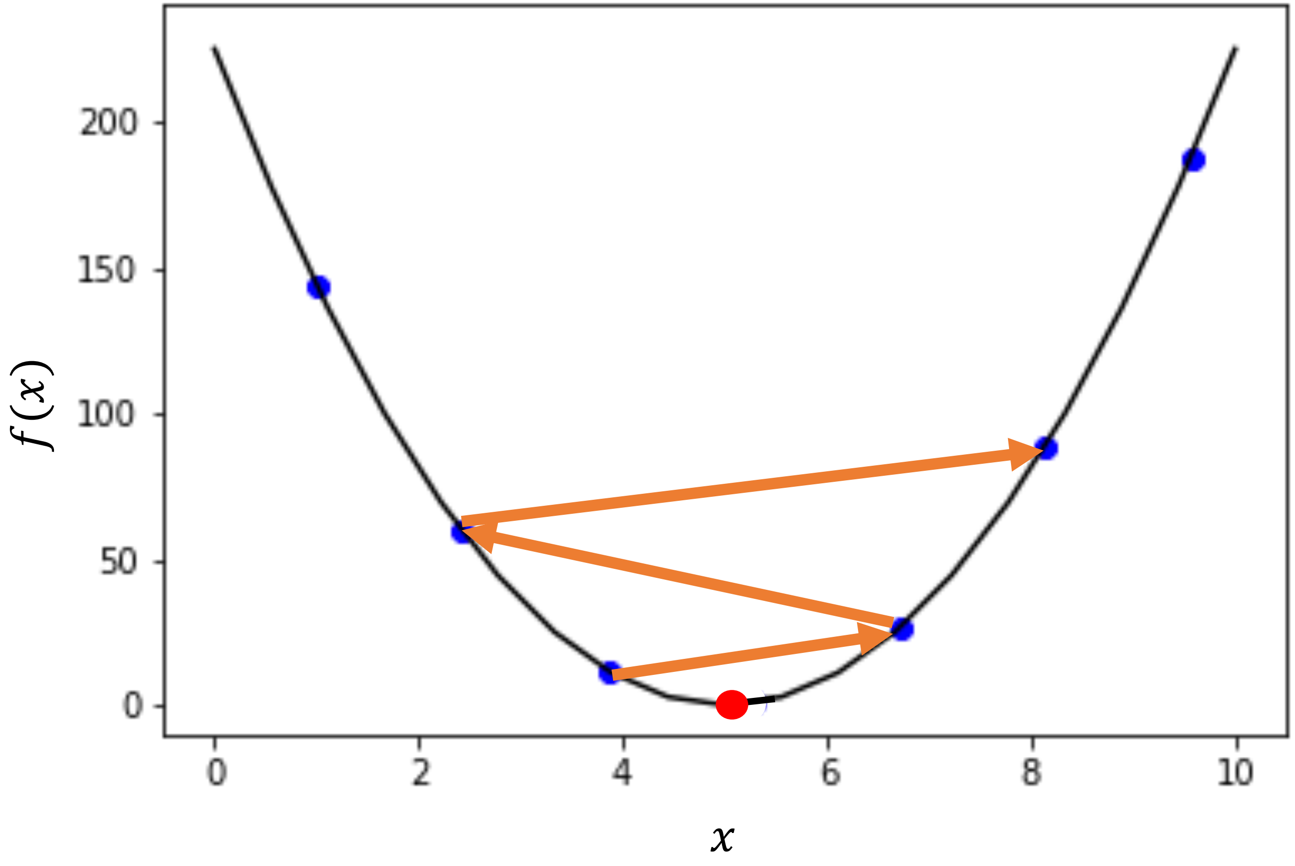

- $\alpha$ grandes: puede no converger.

# Descenso de la pendiente:



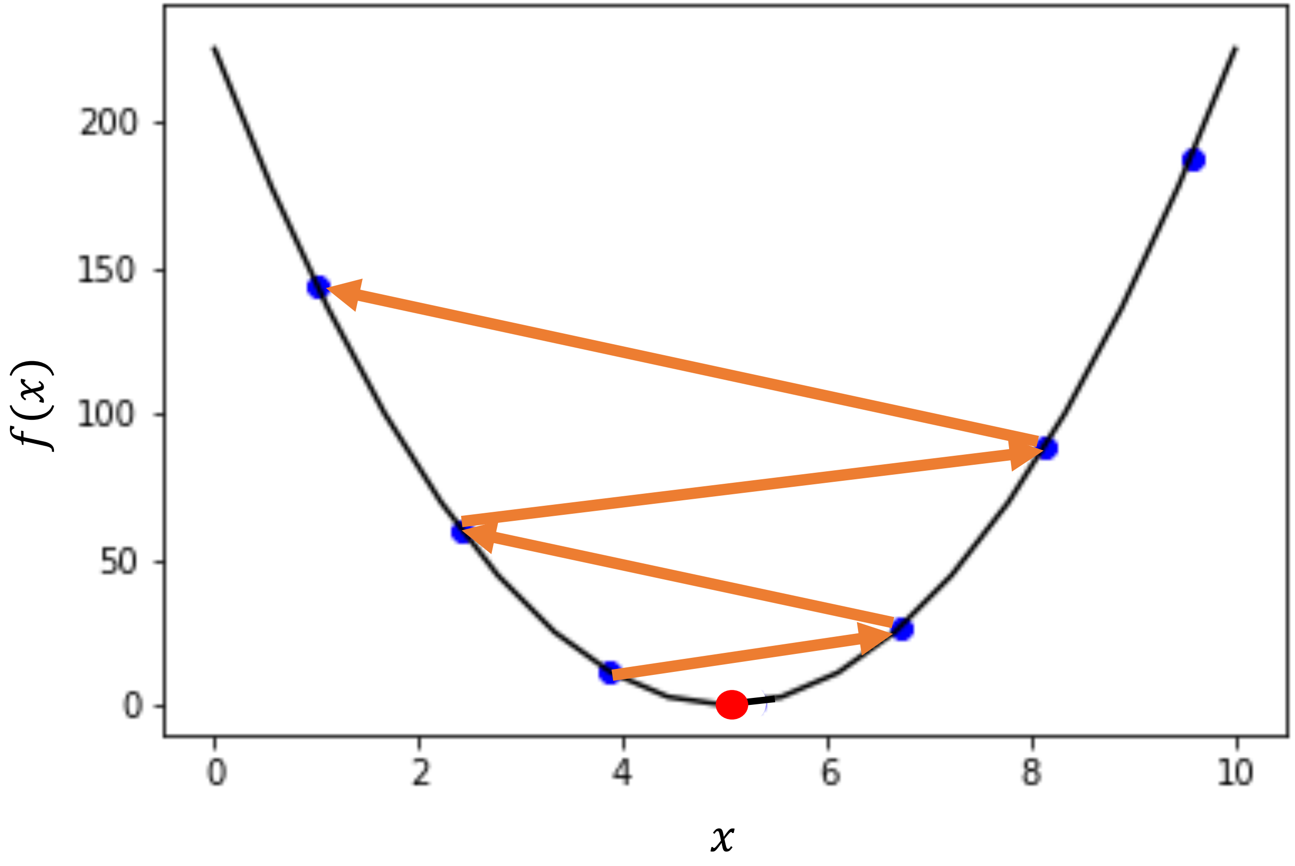

- $\alpha$ grandes: puede no converger.

# Descenso de la pendiente:



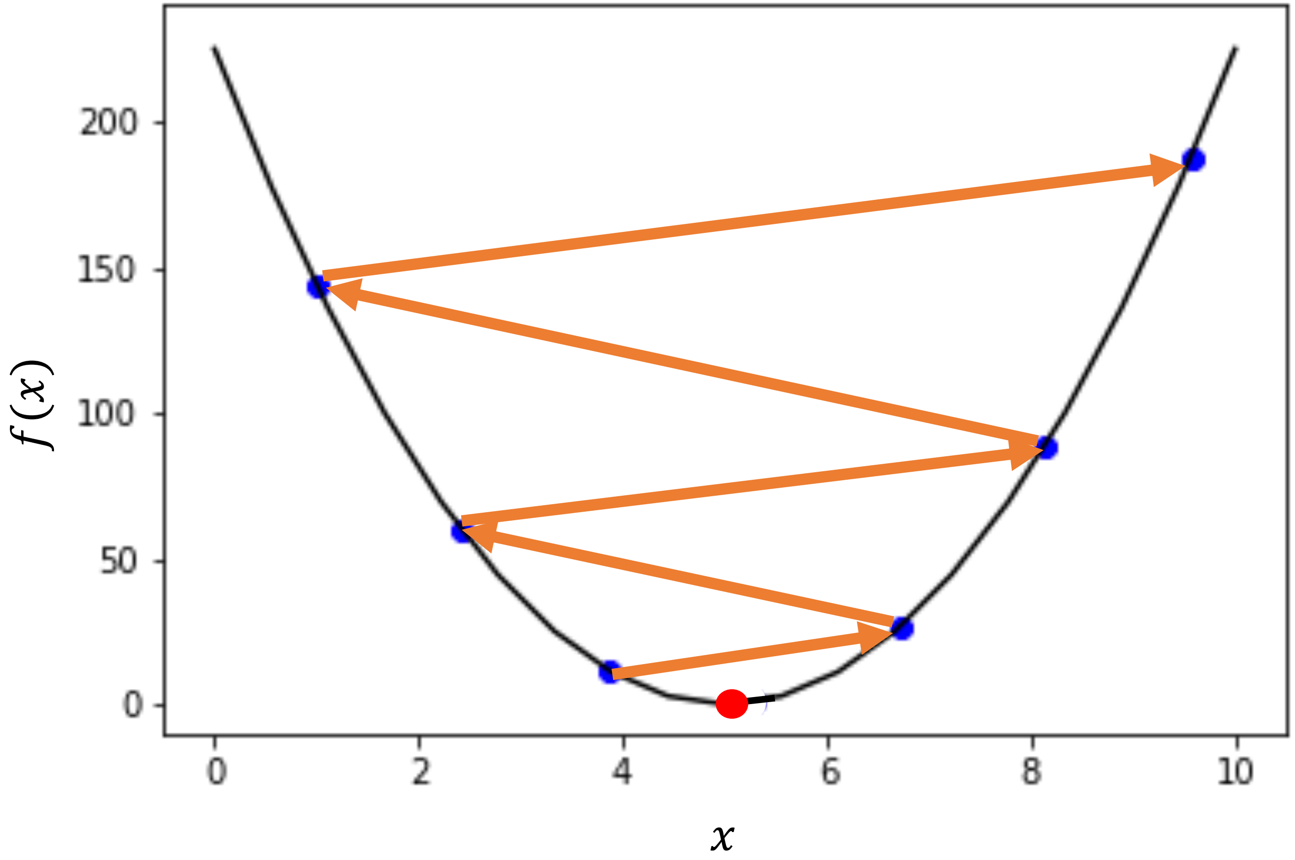

- $\alpha$ grandes: puede no converger.

# Descenso de la pendiente:



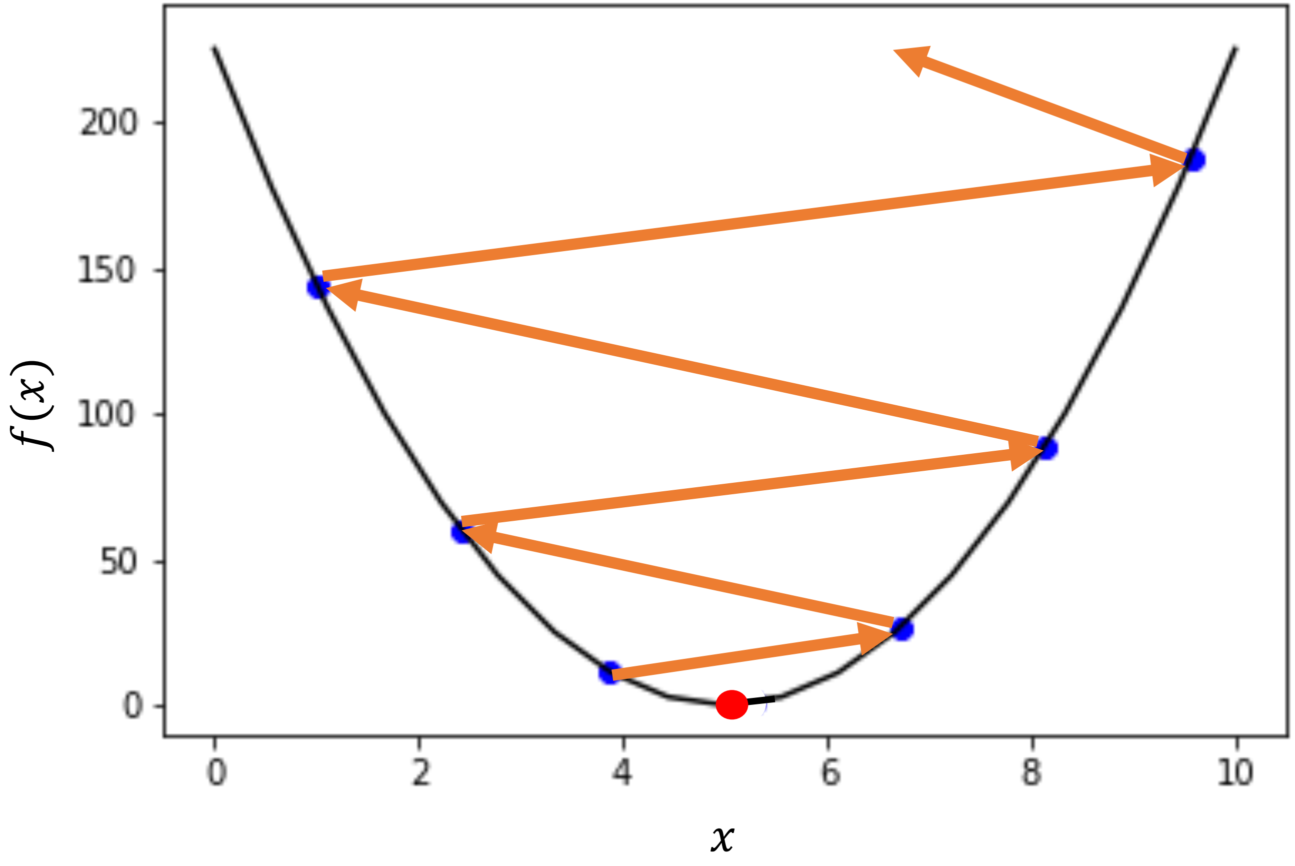

- $\alpha$ grandes: puede no converger.

# Descenso de la pendiente:



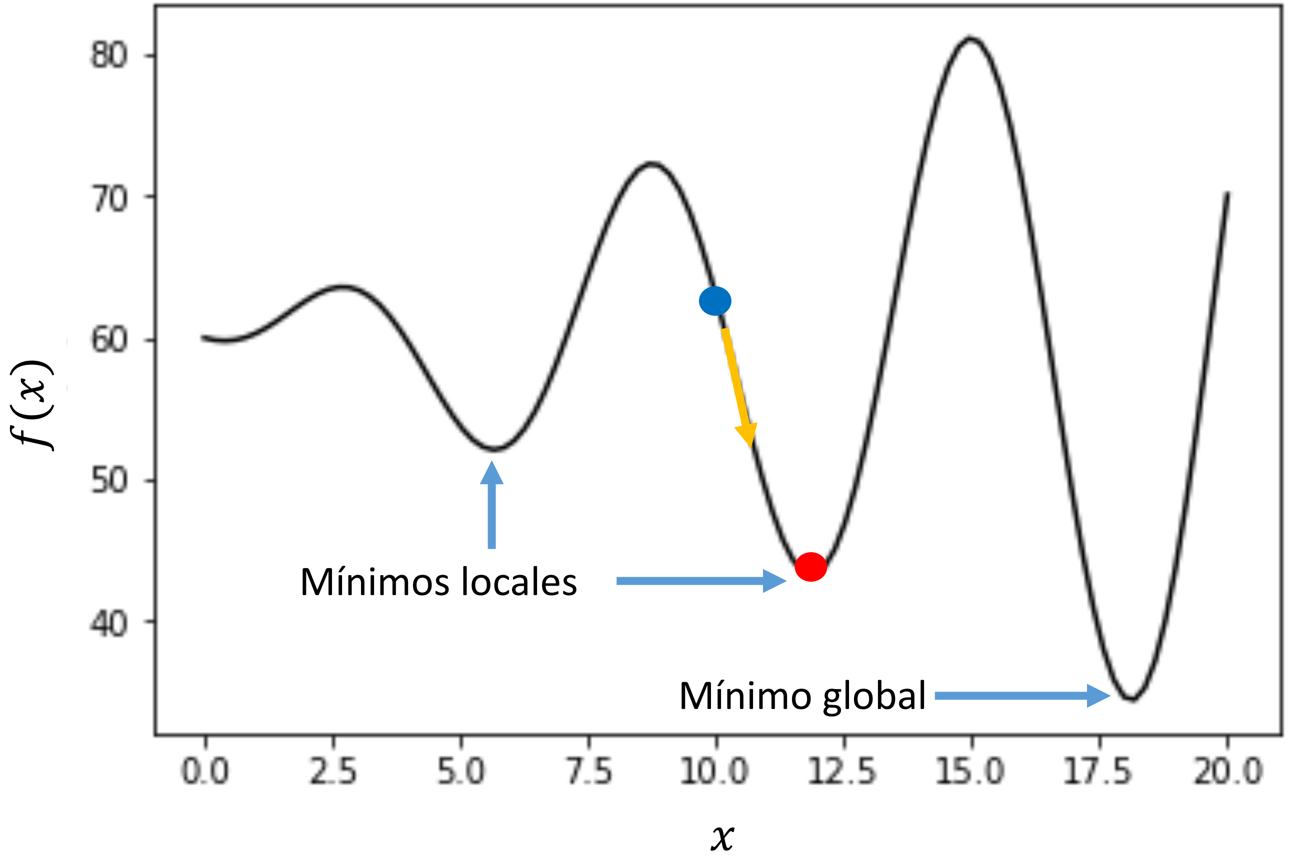

- $\alpha$ grandes: puede no converger.

# Descenso de la pendiente:




- $\alpha$ grandes: puede no converger.

# Descenso de la pendiente:




- Funciones no convexas: podemos caer en mínimos locales.

# Descenso de la pendiente:

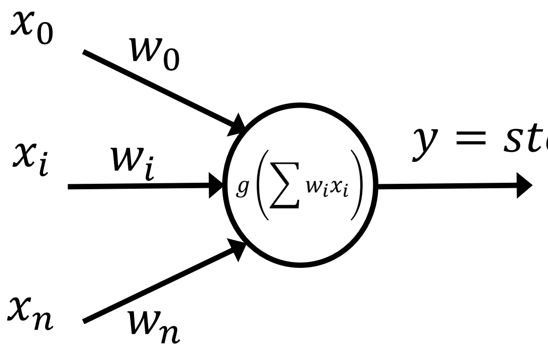
- Gradiente, $f$:
  - Una variable: $\nabla f(x) = \frac{df(x)}{dx}$.
  - Varias variables: $\mathbf{x} = [x_1, x_2, \cdots, x_n] \Rightarrow \nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \cdots \\ \frac{\partial f(\mathbf{x})}{\partial x_2} \end{bmatrix}$.

# Descenso de la pendiente:

- Gradiente, $f$:
  - Una variable: $\nabla f(x) = \frac{df(x)}{dx}$.
  - Varias variables: $\mathbf{x} = [x_1, x_2, \cdots, x_n] \Rightarrow \nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f(\mathbf{r})}{\partial x_1} \\ \cdots \\ \frac{\partial f(\mathbf{r})}{\partial x_n} \end{bmatrix}$.
- En el caso de una neurona artificial, $f$ función perdidas:
  - $\mathbf{w} = [w_1, w_2, \cdots, w_n] \Rightarrow \nabla f(\mathbf{w}) = \begin{bmatrix} \frac{\partial f(\mathbf{w})}{\partial w_1} \\ \cdots \\ \frac{\partial f(\mathbf{w})}{\partial w_2} \end{bmatrix}$.

# Perceptrón:

- Perceptron:
  - Devuelve 1 Devuelve 0 si la suma ponderada de las entradas es menor o igual que un umbral.



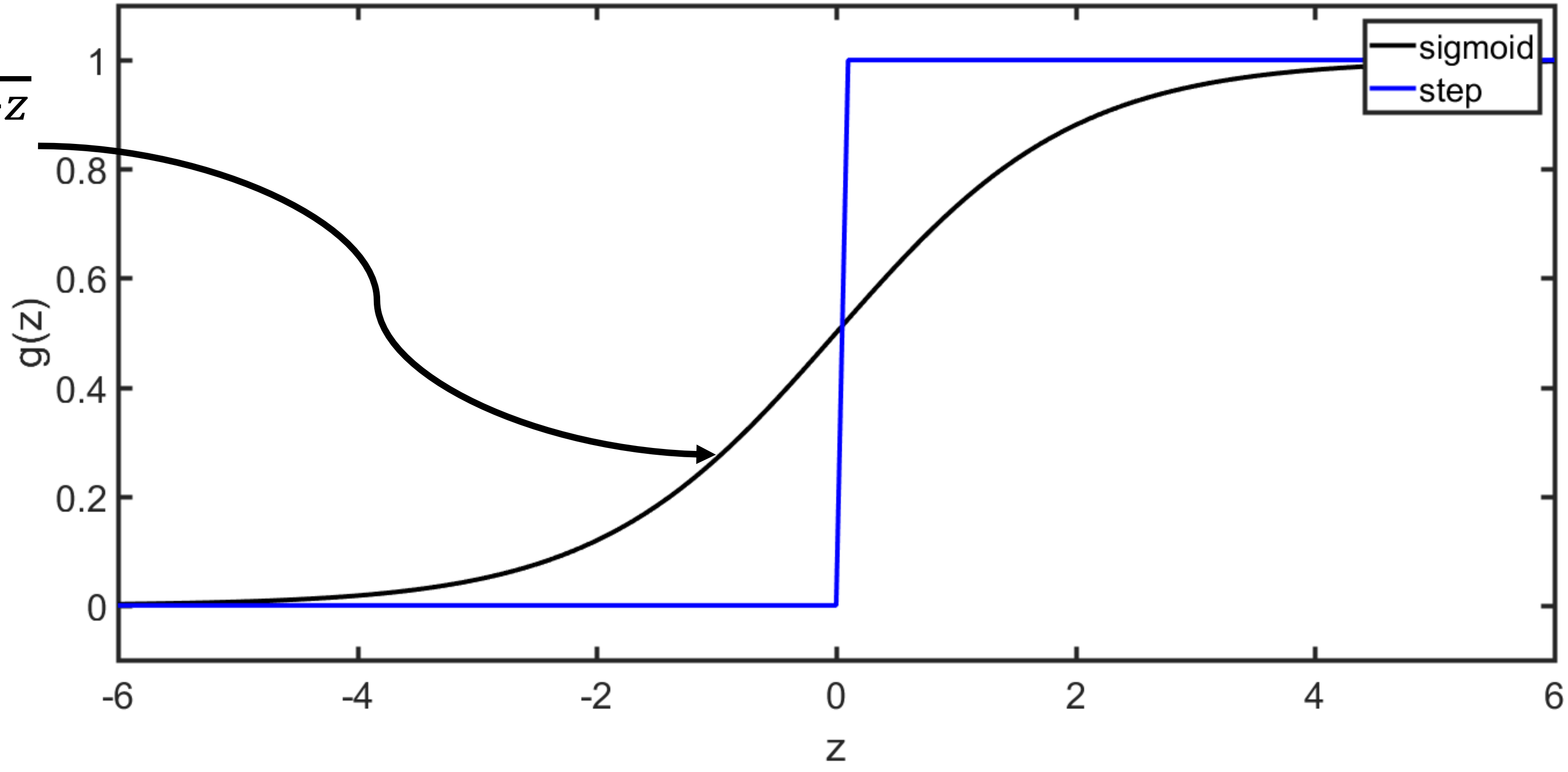

$$y = step\left(\sum w_i x_i\right) = step(w_0 x_0 + \cdots + w_n x_n)$$

  - Problemas del perceptron:
    - $y$ no es diferenciable.
    - $y$ es una función discontinua de las entradas
    - Siempre da una predicción segura que es ‘0’ o ‘1’.

# Perceptrón sigmoide:

- Vamos a utilizar la función sigmoide que es continua y diferenciable:

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



# Perceptrón sigmoide:

- ¿Como actualizamos los coeficientes del perceptrón sigmoide?
  1. Definimos una función de pérdidas: $f = Loss(w_1, \cdots, w_2) = (y_d - y)^2$

# Perceptrón sigmoide:

- ¿Como actualizamos los coeficientes del perceptrón sigmoide?
  1. Definimos una función de pérdidas: $f = Loss(w_1, \cdots, w_2) = (y_d - y)^2$
  2. Utilizando el descenso de la pendiente:
  - $w_i \leftarrow w_i + \beta \frac{\partial}{\partial w_i} Loss(w_1, \cdots, w_2)$

# Perceptrón sigmoide:

- ¿Como actualizamos los coeficientes del perceptrón sigmoide?
  1. Definimos una función de pérdidas: $f = Loss(w_1, \cdots, w_2) = (y_d - y)^2$
  2. Utilizando el descenso de la pendiente:
  - $w_i \leftarrow w_i + \beta \frac{\partial}{\partial w_i} Loss(w_1, \cdots, w_2)$
    - $\frac{\partial}{\partial w_i} Loss(w_1, \cdots, w_2) = \frac{\partial}{\partial w_i}(y_d - y)^2 = 2(y_d - y)\frac{\partial}{\partial w_i}(y_d - y) = -2(y_d - y)\frac{\partial y}{\partial w_i}.$

# Perceptrón sigmoide:

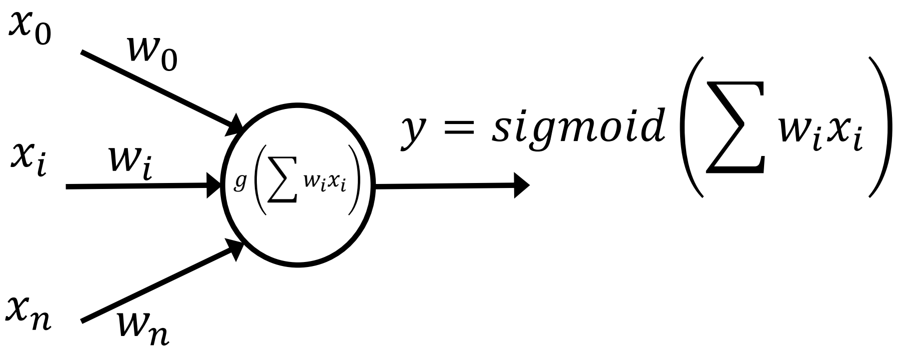
- ¿Como actualizamos los coeficientes del perceptrón sigmoide?
  1. Definimos una función de pérdidas: $f = Loss(w_1, \cdots, w_2) = (y_d - y)^2$
  2. Utilizando el descenso de la pendiente:
  - $w_i \leftarrow w_i + \beta \frac{\partial}{\partial w_i} Loss(w_1, \cdots, w_2)$
    - $\frac{\partial}{\partial w_i} Loss(w_1, \cdots, w_2) = \frac{\partial}{\partial w_i} (y_d - y)^2 = 2(y_d - y) \frac{\partial}{\partial w_i} (y_d - y) = -2(y_d - y) \frac{\partial y}{\partial w_i}.$
    - $y = \frac{1}{1+e^{-\sum w_i x_i}} \Rightarrow \frac{\partial y}{\partial w_i} = y(1-y)\, x_i.$

# Perceptrón sigmoide:

- ¿Como actualizamos los coeficientes del perceptrón sigmoide?
  1. Definimos una función de pérdidas: $f = Loss(w_1, \cdots, w_2) = (y_d - y)^2$
  2. Utilizando el descenso de la pendiente:
  - $w_i \leftarrow w_i + \beta \frac{\partial}{\partial w_i} Loss(w_1, \cdots, w_2)$
    - $\frac{\partial}{\partial w_i} Loss(w_1, \cdots, w_2) = \frac{\partial}{\partial w_i}(y_d - y)^2 = 2(y_d - y)\frac{\partial}{\partial w_i}(y_d - y) = -2(y_d - y)\frac{\partial y}{\partial w_i}.$
    - $y = \frac{1}{1+e^{-\sum w_i x_i}} \Rightarrow \frac{\partial y}{\partial w_i} = y(1-y)\, x_i.$
    - $\frac{\partial}{\partial w_i} Loss(w_1, \cdots, w_2) = -2(y_d - y)y(1-y)x_i.$

# Perceptrón sigmoide:

- ¿Como actualizamos los coeficientes del perceptrón sigmoide?
    1. Definimos una función de pérdidas: $f = Loss(w_1, \cdots, w_2) = (y_d - y)^2$
    2. Utilizando el descenso de la pendiente:
    - $w_i \leftarrow w_i + \beta \frac{\partial}{\partial w_i} Loss(w_1, \cdots, w_2)$
        - $\frac{\partial}{\partial w_i} Loss(w_1, \cdots, w_2) = \frac{\partial}{\partial w_i}(y_d - y)^2 = 2(y_d - y)\frac{\partial}{\partial w_i}(y_d - y) = -2(y_d - y)\frac{\partial y}{\partial w_i}.$
        - $y = \frac{1}{1+e^{-\sum w_i x_i}} \Rightarrow \frac{\partial y}{\partial w_i} = y(1-y)\, x_i.$
        - $\frac{\partial}{\partial w_i} Loss(w_1, \cdots, w_2) = -2(y_d - y)y(1-y)x_i.$
    - $w_i \leftarrow w_i + \alpha\, (y_d - y)y(1-y)x_i.$



# Perceptrón sigmoide:

- Entrenamiento:
  1. Se asignan pesos aleatorios a las entradas:
     - $w_i \in (-0.5,\ 0.5)$.

# Perceptrón sigmoide:

- Entrenamiento:
  1. Se asignan pesos aleatorios a las entradas:
     - $w_i \in (-0.5,\ 0.5)$.
  2. Se presenta el siguiente conjunto de datos de entrenamiento en las entradas y se observa la salida.

# Perceptrón sigmoide:

- Entrenamiento:
  1. Se asignan pesos aleatorios a las entradas:
     - $w_i \in (-0.5,\ 0.5)$.
  2. Se presenta el siguiente conjunto de datos de entrenamiento en las entradas y se observa la salida.
  3. Si la salida es incorrecta se cambian los valores de los pesos de acuerdo con la expresión:
     - $w_i \leftarrow w_i + \alpha\,(y_d - y)y(1 - y)x_i$.

# Perceptrón sigmoide:

- Entrenamiento:
  1. Se asignan pesos aleatorios a las entradas:
     - $w_i \in (-0.5,\ 0.5)$.
  2. Se presenta el siguiente conjunto de datos de entrenamiento en las entradas y se observa la salida.
  3. Si la salida es incorrecta se cambian los valores de los pesos de acuerdo con la expresión:
     - $w_i \leftarrow w_i + \alpha\,(y_d - y)y(1 - y)x_i$.
  - Se repite el procedimiento hasta que el error esté por debajo del deseado.