Descenso de gradiente

Entrenamiento = Aprendizaje = Optimización

Aprendizaje Supervisado

- Función de error E
- Ejemplos con:

  Valores de entrada xi

  Valores de salida yi

Buscar parámetros óptimos en base a ejemplos y E

Datos

estudio	nota
2	1
5	3.2
7	4.5
9	6
10	4

Aprendizaje
(optimización de E)

Modelo generado

m y *b*

Optimización de funciones

Optimizar= buscar configuración de parámetros para que la función sea mínima (o máxima).

Función convexa

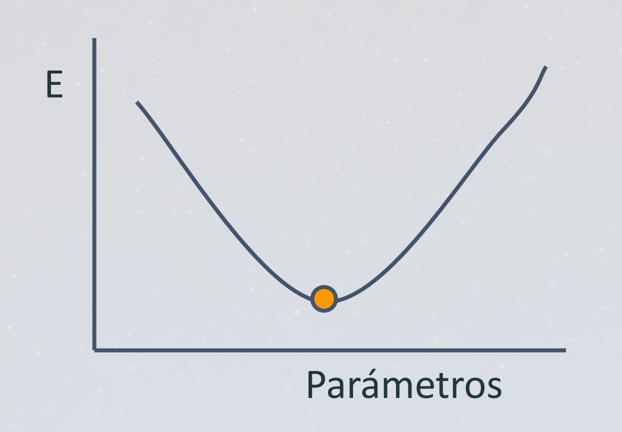
Único mínimo (global)

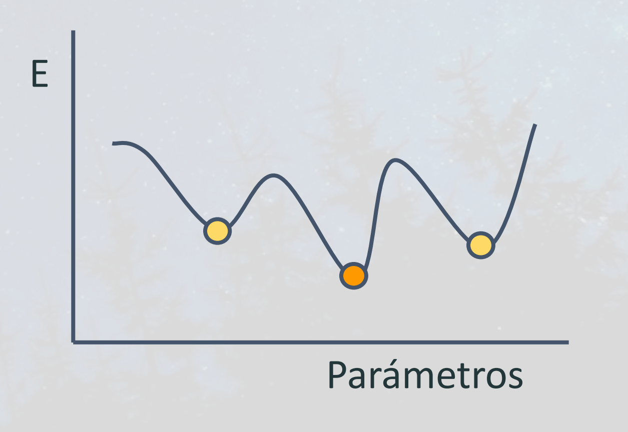
Error para Regresión Lineal

Función no convexa

Muchos mínimo locales

Error para Redes Neuronales





Algoritmos de Optimización de funciones

Generales

Pocas asunciones sobre f - Pocas garantías – Mayor tiempo

Ejemplos: Fuerza bruta, Búsqueda aleatoria, Metaheurísticas.

Especializados

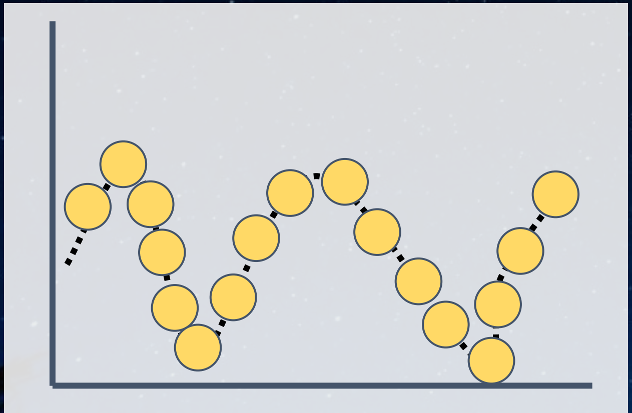
Muchas asunciones sobre f – Mayor garantía. No siempre existen.

Ejemplo: camino mínimo en un grafo

Algoritmos de Optimización de funciones

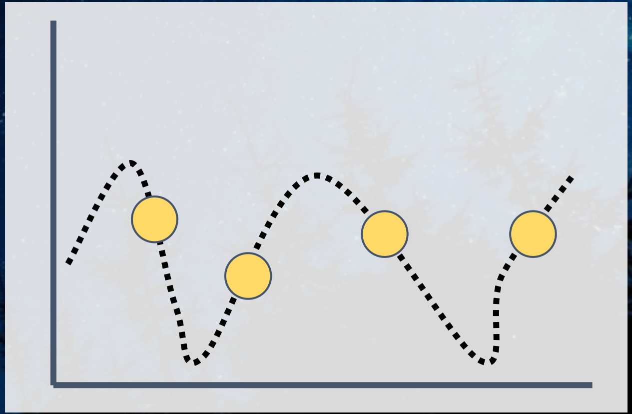
Fuerza bruta

- Probar todos los valores posibles
- Quedarse con el mejor
  - ¿Parámetros continuos?
  - ¿Muchos parámetros?



Búsqueda aleatoria

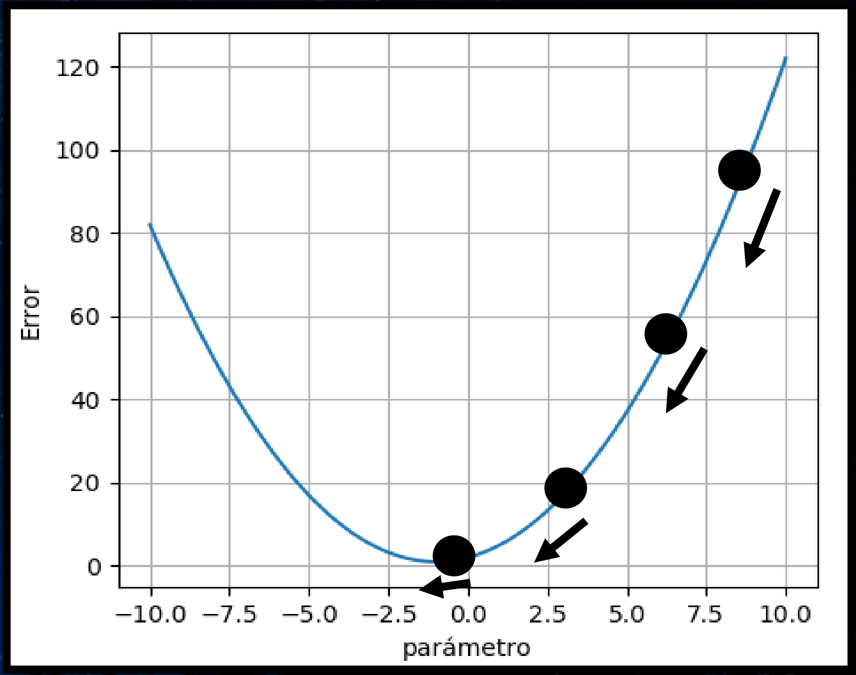
- Evaluar valores aleatorios
- Quedarse con el mejor
- ¿Cómo generar valores aleatorios?



Descenso de Gradiente

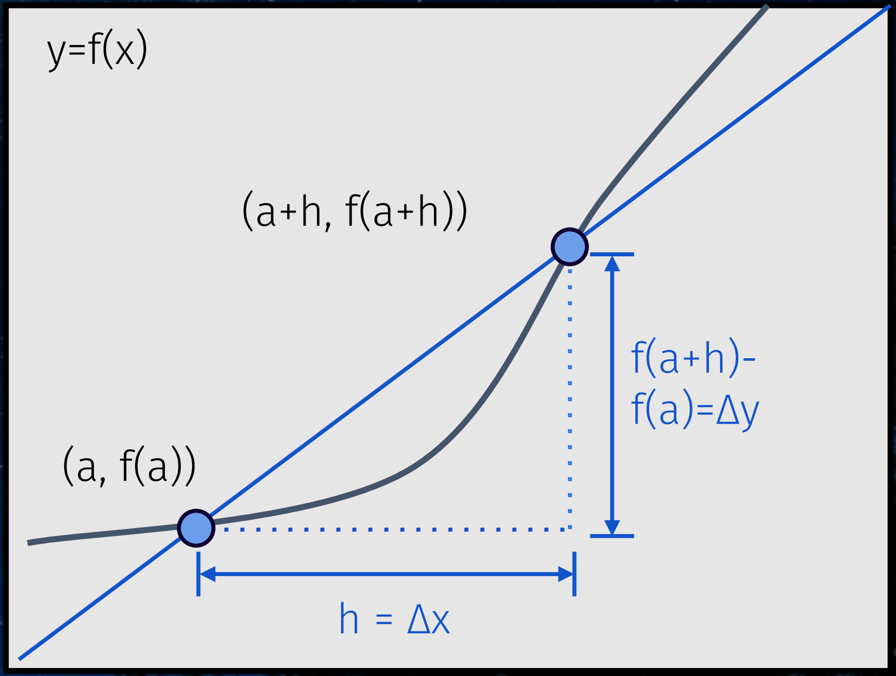
Algoritmo general

- Asunción: f es derivable (Puede tener puntos de discontinuidad).
- Utilizar el gradiente o derivada para guiar la optimización.
- Sensible a mínimos locales.
- Eficiente para muchos datos.
- Generalizable: Redes neuronales, SVM, etc.



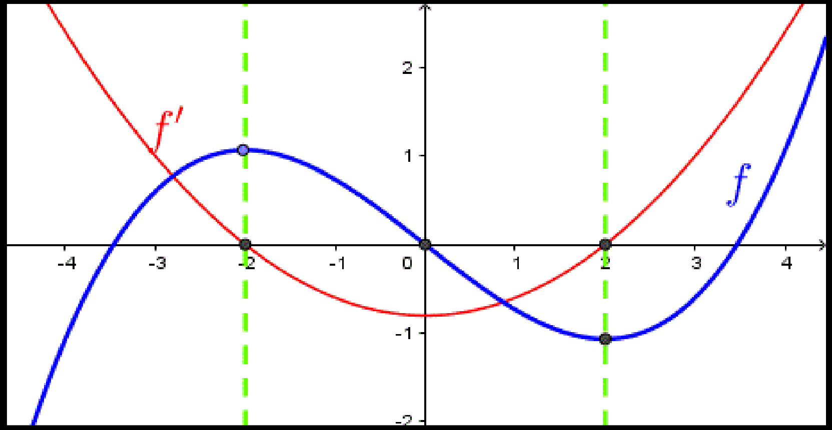
Derivadas

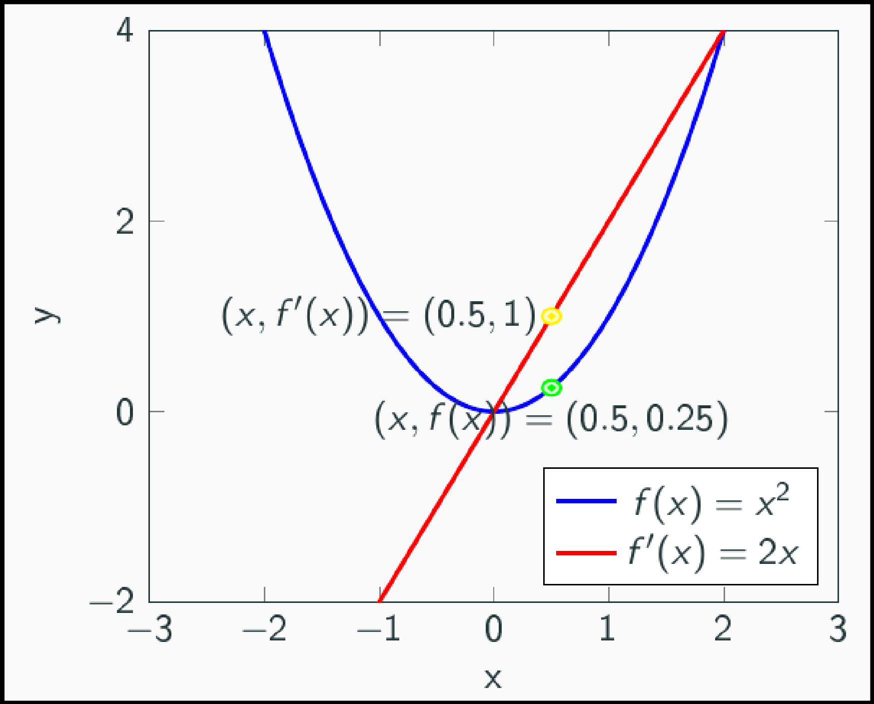
- **La derivada** de la función en un punto es equivalente a la pendiente de la recta tangente.
- Equivalente a la razón de cambio= $\Delta y/\Delta x$.
- El signo indica la dirección de crecimiento.



Derivadas

$$f'(x) = lim_{h\to 0}\frac{f(x+h)-f(x)}{h} = \begin{cases} >0 & \text{si } f \text{ crece} \\ <0 & \text{si } f \text{ decrece} \\ =0 & \text{pto crítico} \end{cases}$$





Descenso de gradiente en 1D

Iterativamente calcular la derivada del **Error** y recalcular $\boldsymbol{w}$ con un factor de ajuste $\boldsymbol{\alpha}$.

Dados:

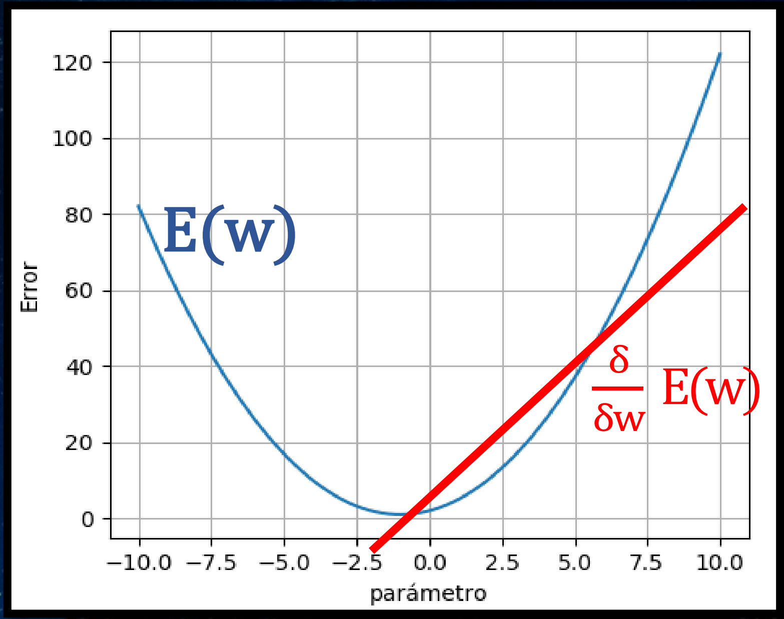
Función de Error E

Parámetro w

Velocidad de aprendizaje α

Iterar hasta converger:

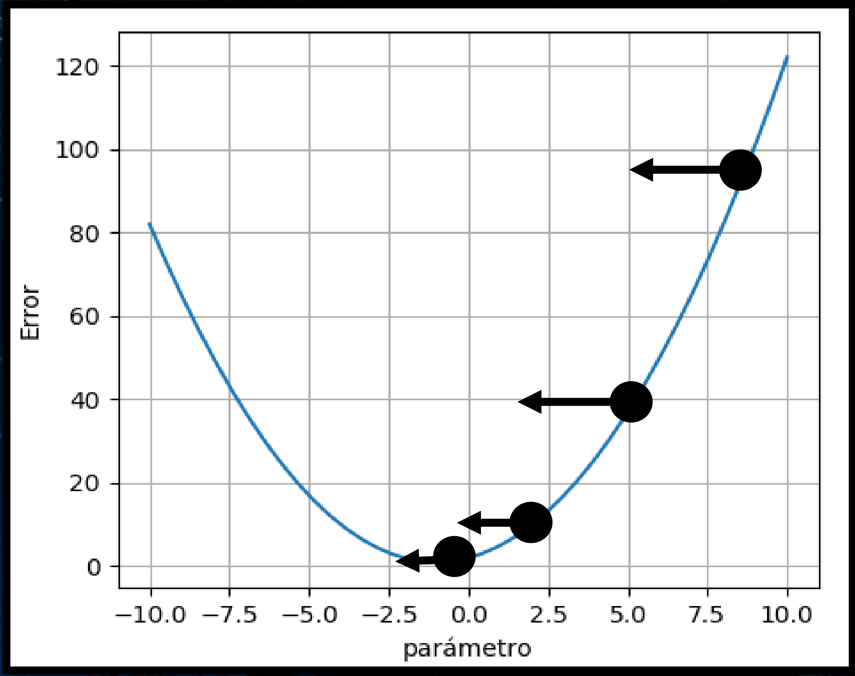
$$w = w - \alpha \frac{\delta}{\delta w} E(w)$$



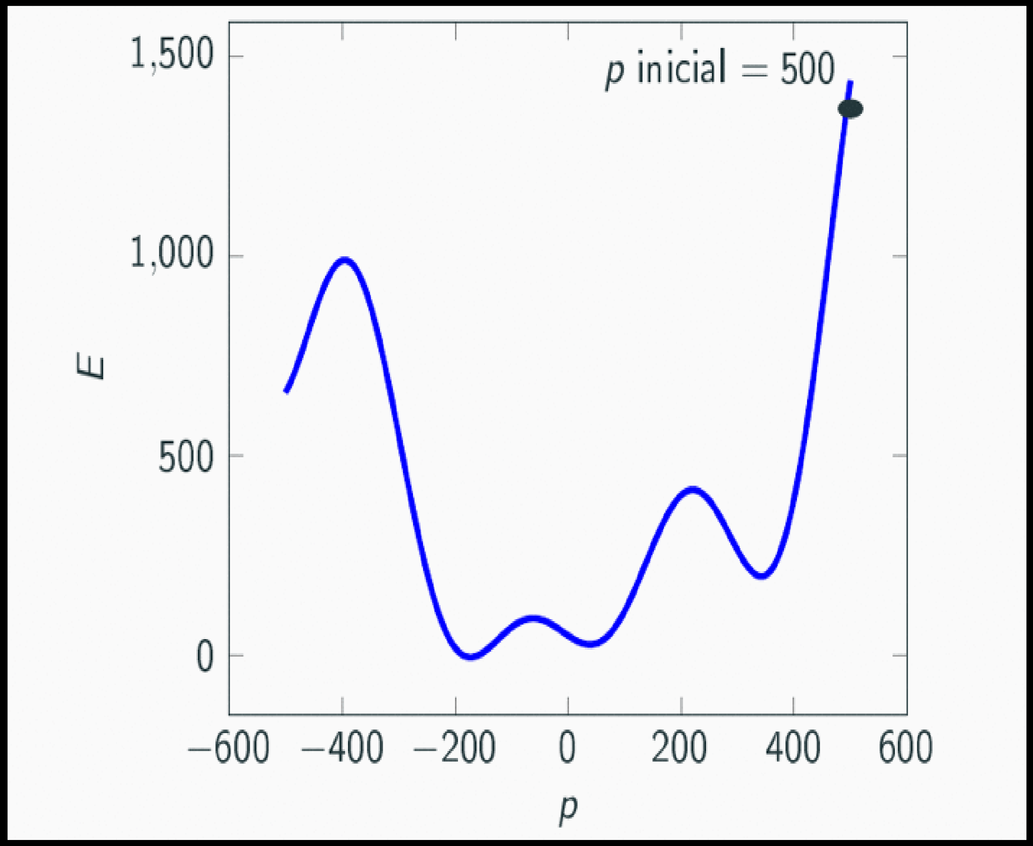
Descenso de gradiente en 1D

Iterar hasta converger:

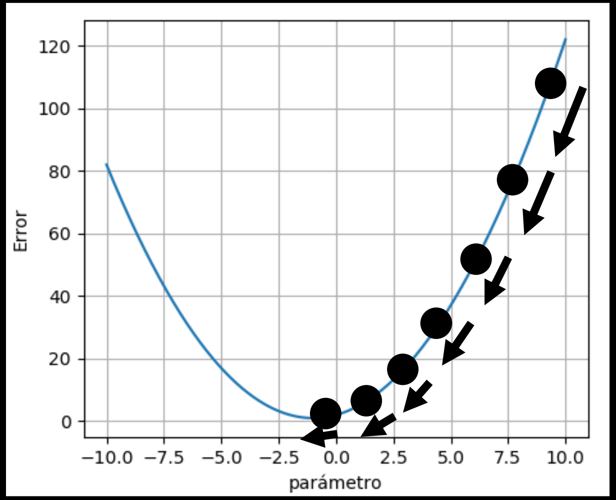
$$w = \underbrace{w}_{\bullet} - \underbrace{\alpha \frac{\delta}{\delta w} E(w)}_{\longrightarrow}$$

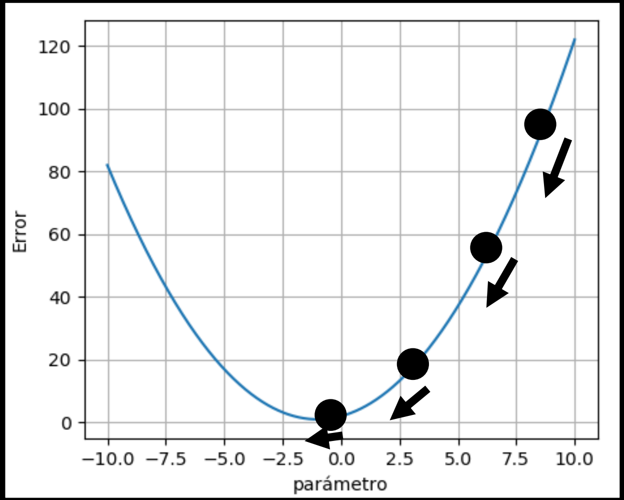


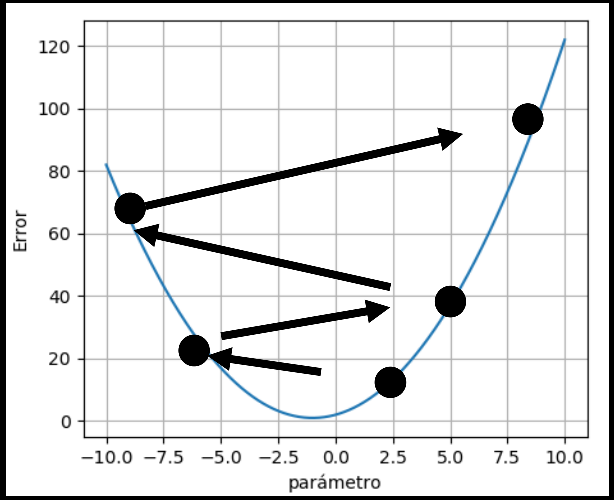
Descenso de gradiente en 1D



Descenso de gradiente en 1D. Alfa.







α muy pequeño,
- Poco avance por iteración
- Alto costo computacional

α “correcto”,
- Buen avance por iteración
- Costo computacional correcto.

α muy grande,
- Salto grandes. Puede divergir.
- Problemas numéricos.

Descenso de gradiente en 2D

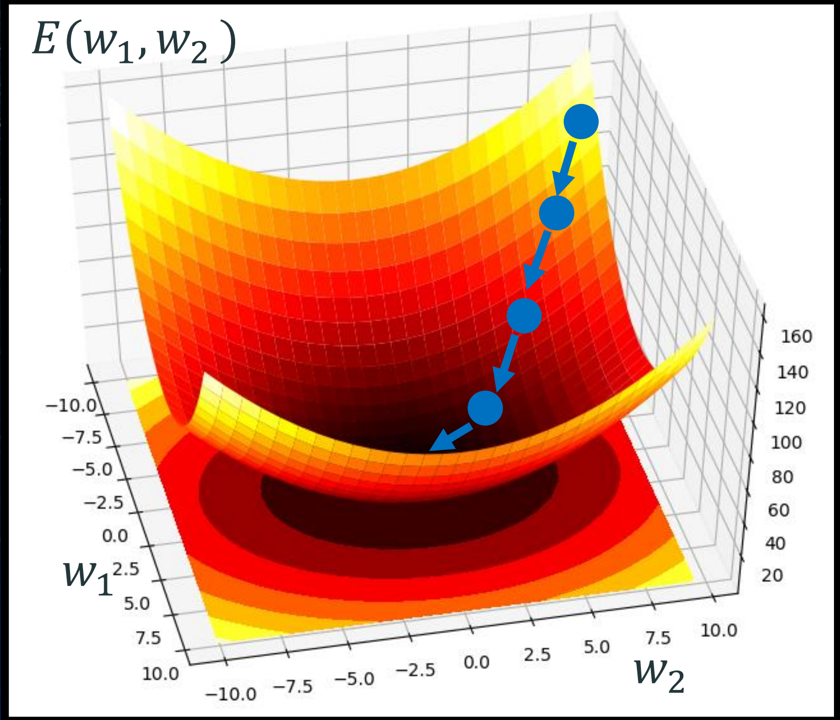
Descenso de gradiente en 2D

- Mismo algoritmo
- 2 parámetros
- $E(w_1, w_2)$,
- ΔE = (δE/δw1,δE/δw2)

Hasta converger:

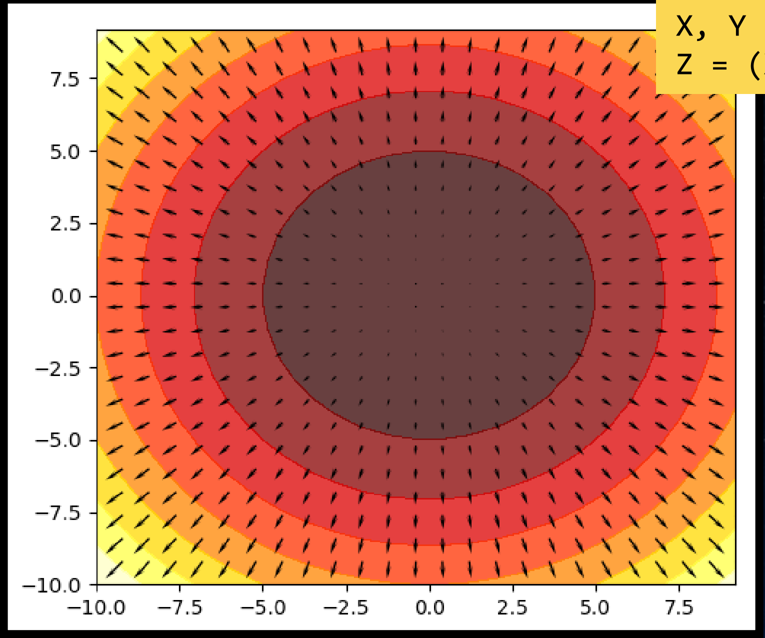
$$w_i = w_i - \alpha \frac{\delta}{\delta w_i} E(w_1, w_2)$$

$$para\ i = \{1,2\}$$

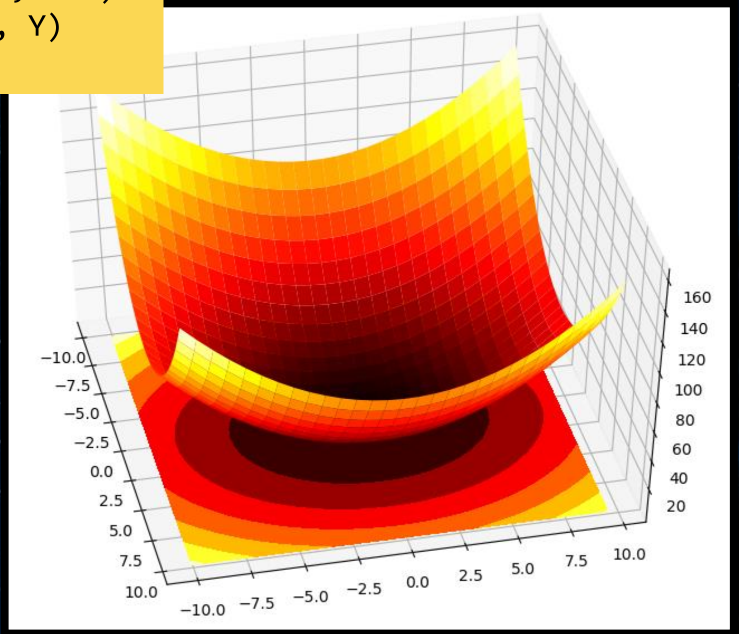


Descenso de gradiente en 2D

```
X = np.arange(-10, 10, 0.8)
Y = np.arange(-10, 10, 0.8)
X, Y = np.meshgrid(X, Y)
Z = (X**2 + Y**2)
```




```
plt.contour(X, Y, Z)
dhdy, dhdx = np.gradient(Z)
plt.quiver(X, Y, dhdx, dhdy)
```

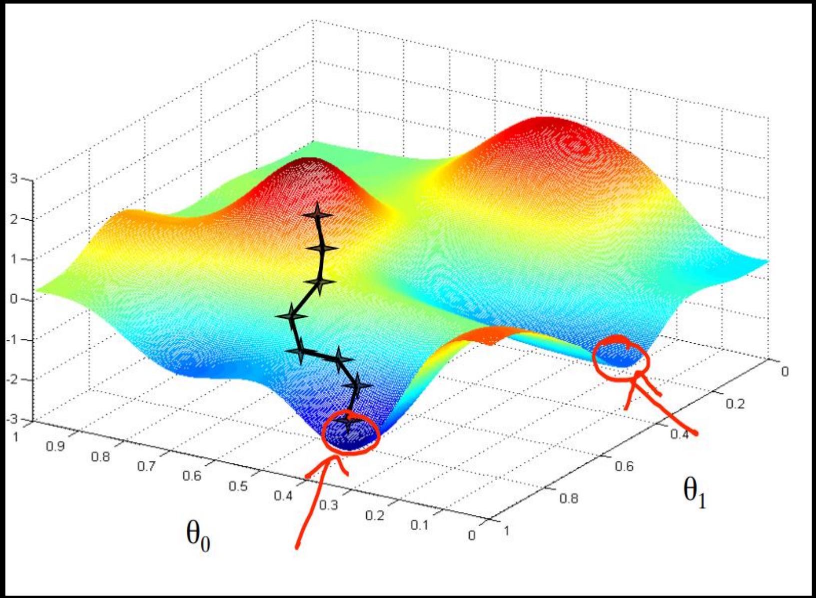


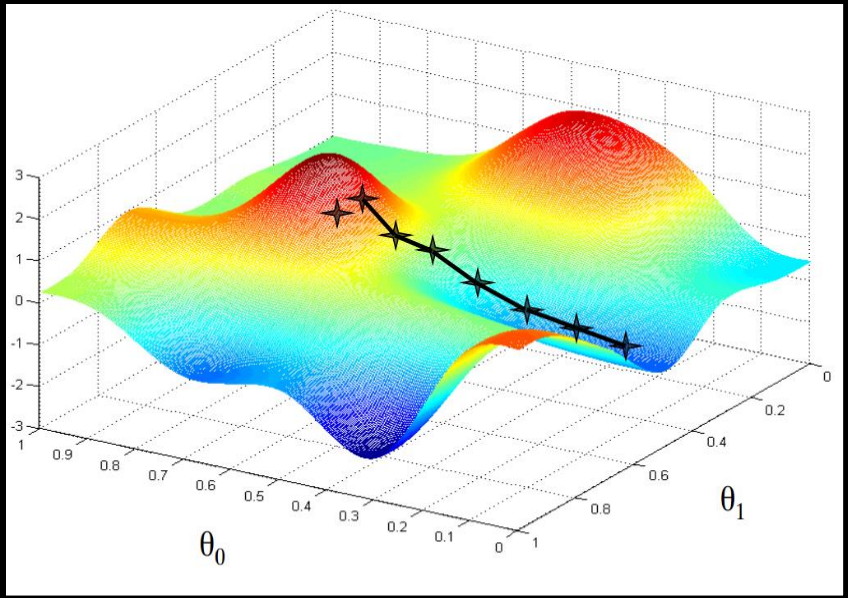

```
ax = Axes3D(fig)
ax.plot_surface(X, Y, Z)
ax.contourf(X, Y, Z)
```

Funciones no convexas

Funciones no convexas (ej. Redes Neuronales)

- Como elegir valores iniciales de parámetros.
- Mínimos locales





Resumen

Descenso de gradiente

- Iterativo
- Generalizable
- Requiere que E sea diferenciable.
- Escalable a muchos datos.
- Ecuación de cambio:
  - $w_i = w_i - \alpha \frac{\delta}{\delta w_i} E(w_1 \dots w_n)$