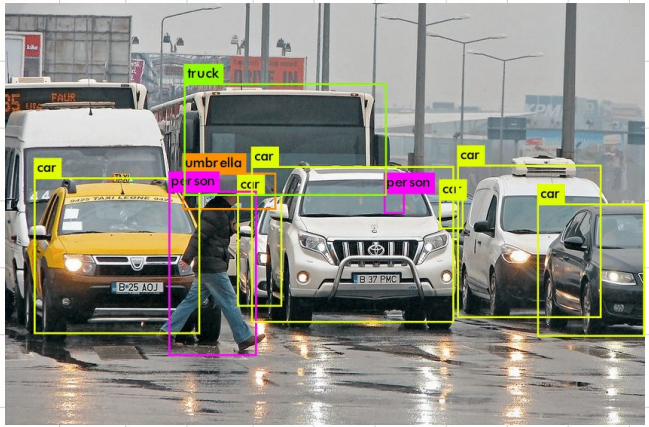
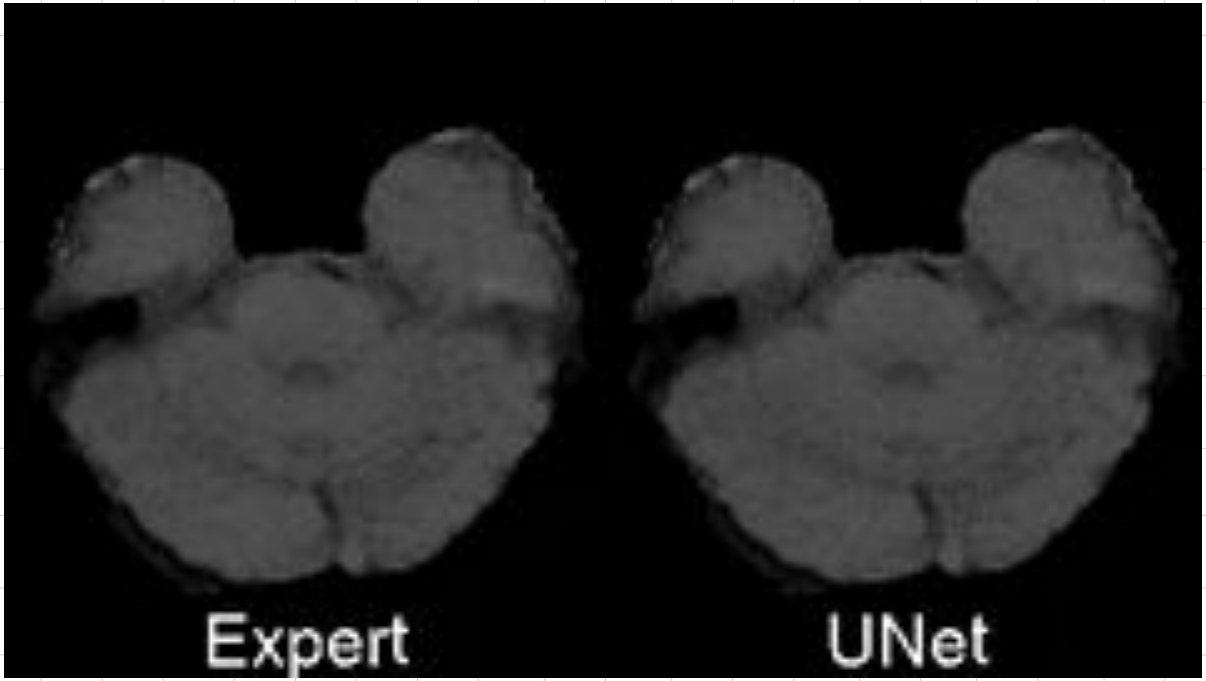
Descenso de gradiente

Autor: Sanchez Sauñe, Cristhian Wiki

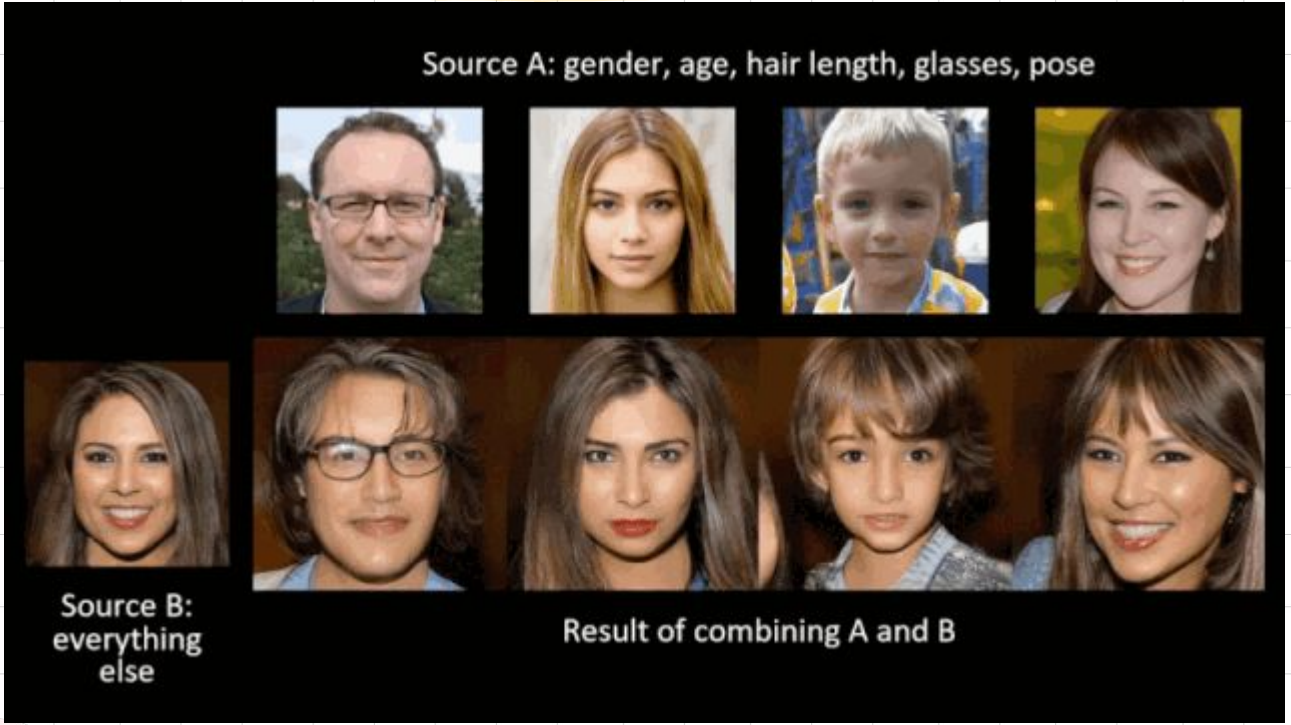
Aplicaciones



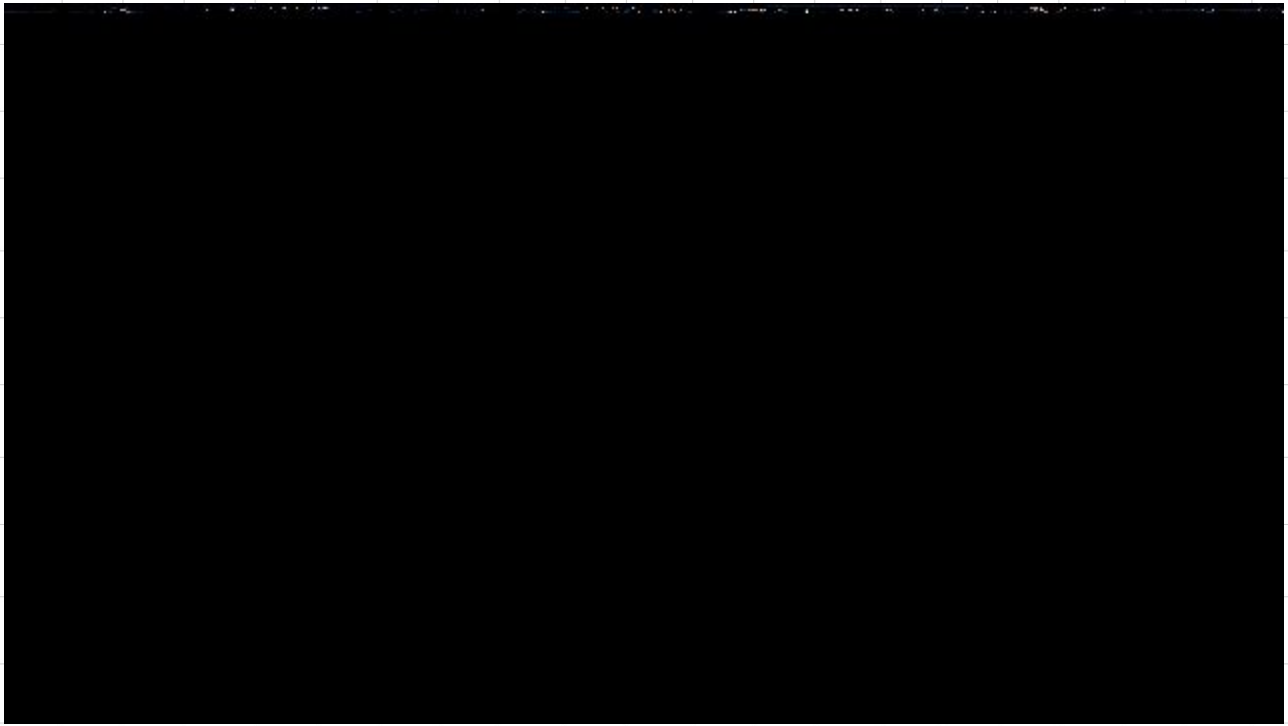
Aplicaciones



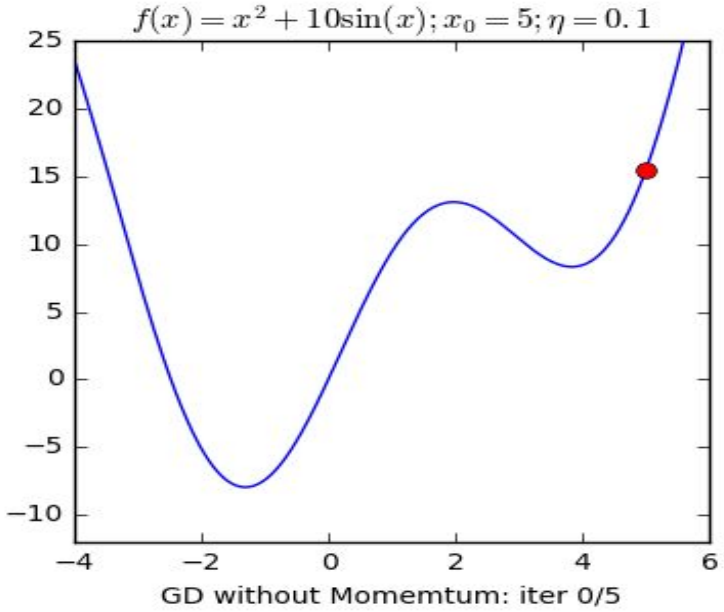
Aplicaciones



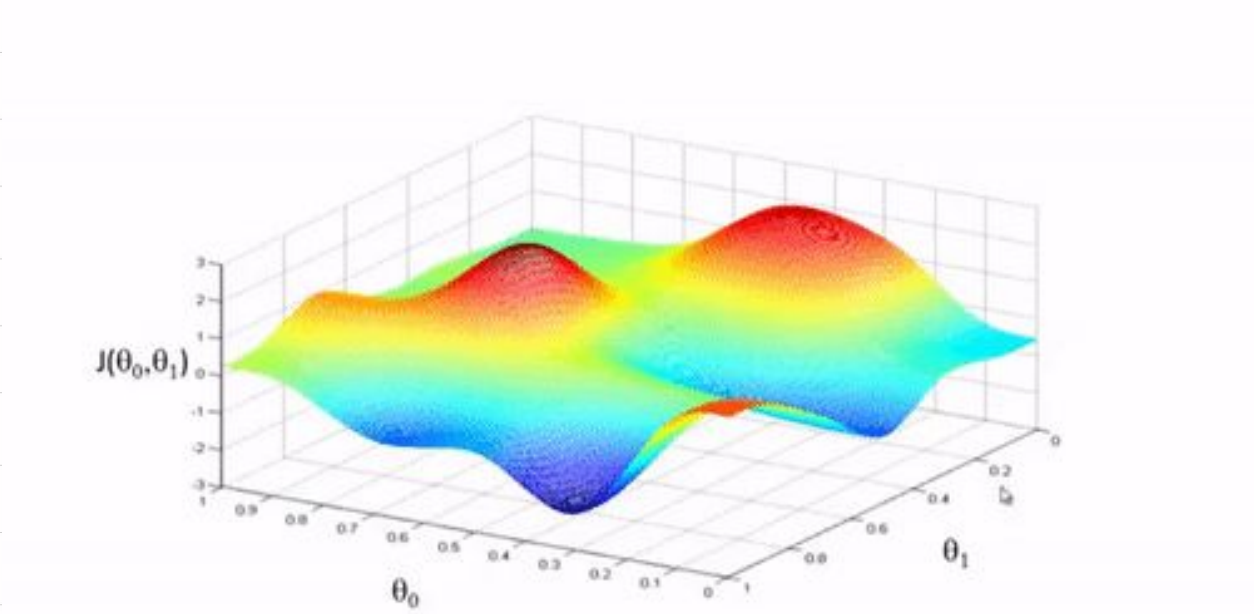
Aplicaciones



Intuición


$f(x) = x^2 + 10\sin(x); x_0 = 5; \eta = 0.1$
25
20
15
10
5
0
−5
−10
−4
−2
0
2
4
6
GD without Momemtum: iter 0/5

Descenso de gradiente en acción



Un problema de optimización

Problema:

$$\min_{w} f(w)$$

Solución Iterativa:

$$w_{k+1} = w_k - \lambda_k \nabla f(w_k)$$

donde,

- w_{k+1} es el valor actualizado luego de k iteraciones
- w_k es el valor inicial antes de la iteración k-ésima,
- λ_k es el tamaño de paso,
- $\nabla f(w_k)$ es el gradiente de f.

Algoritmo completo

Cost Function

$$J(\Theta_0, \Theta_1) = \frac{1}{2m}\sum_{i=1}^{m}[h_\Theta(x_i) - y_i]^2$$

Predicted Value — True Value

Gradient Descent

$$\Theta_j = \Theta_j - \alpha\frac{\partial}{\partial\Theta_j}J(\Theta_0, \Theta_1)$$

Learning Rate

Now,

$$\begin{aligned}\frac{\partial}{\partial\Theta}J_\Theta &= \frac{\partial}{\partial\Theta}\frac{1}{2m}\sum_{i=1}^{m}[h_\Theta(x_i) - y]^2 \\ &= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x_i) - y)\frac{\partial}{\partial\Theta_j}(\Theta x_i - y) \\ &= \frac{1}{m}(h_\Theta(x_i) - y)x_i\end{aligned}$$

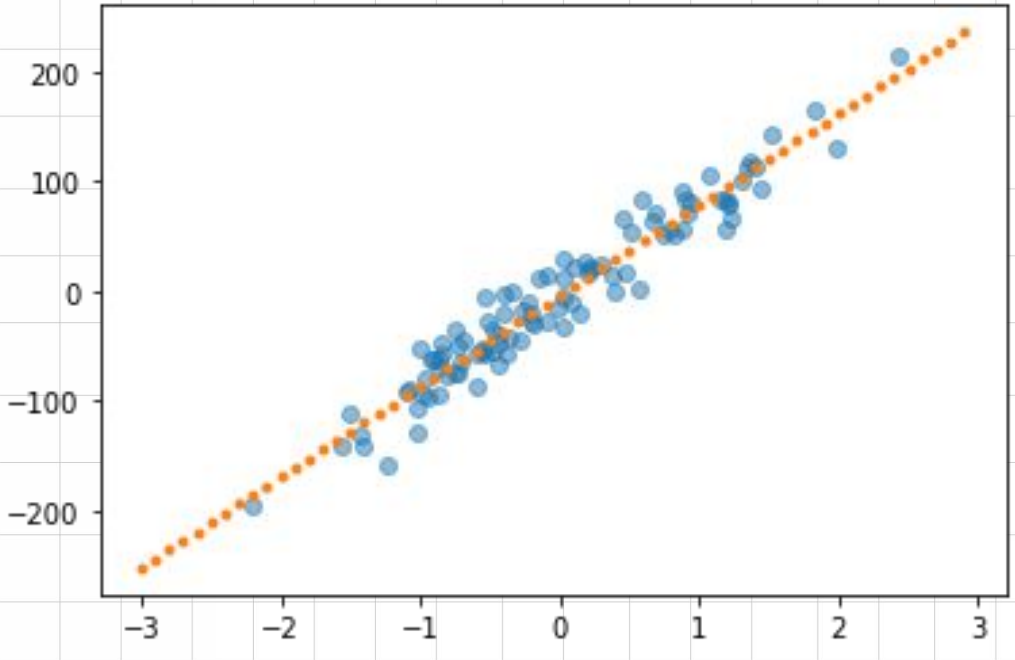
Therefore,

$$\Theta_j := \Theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}[(h_\Theta(x_i) - y)x_i]$$

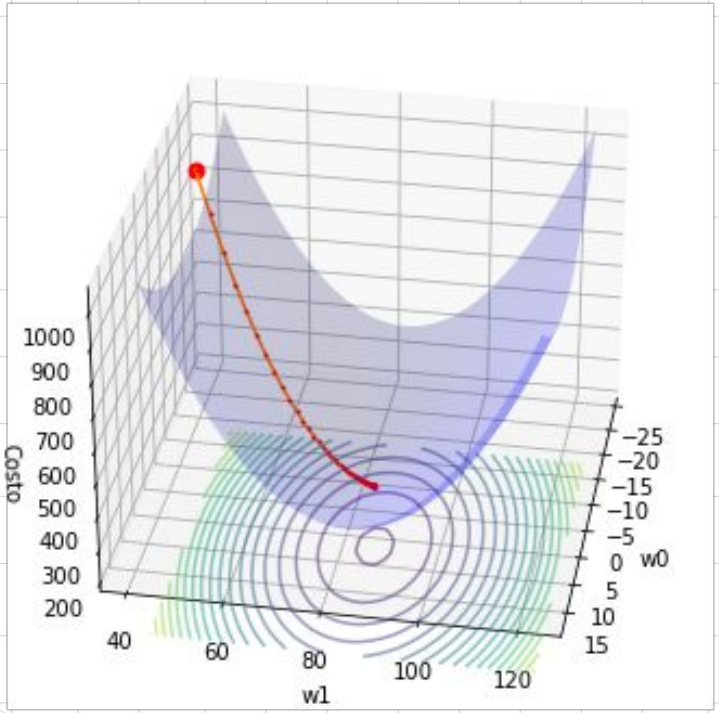
Código

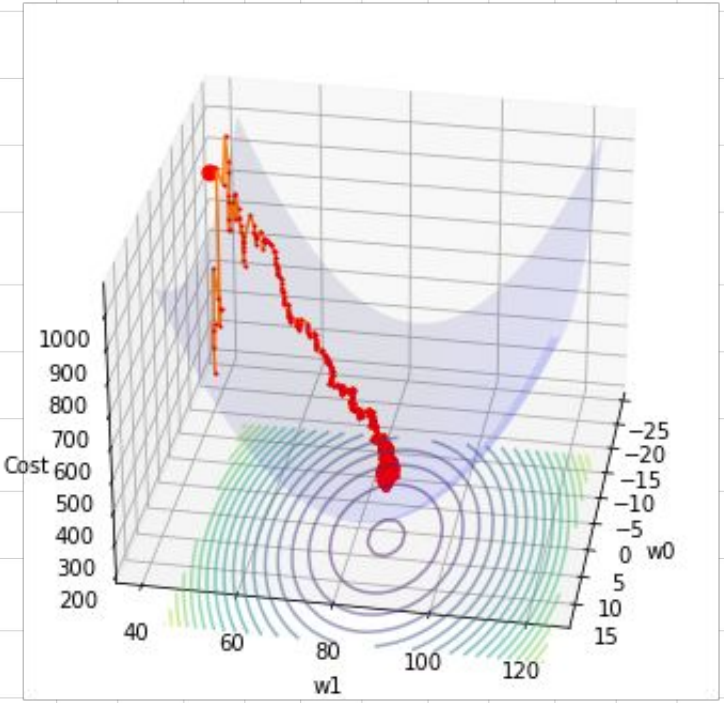
```
def gradient_descent(X,y,theta,learning_rate=0.01,iterations=100):
    '''
    X    = Matriz de X con unidades de sesgo agregadas
    y    = Vector de Y
    theta=Vector de thetas np.random.randn(j,1)
    learning_rate
    iterations = número de iteraciones

    Devuelve el vector theta final y la matriz del historial de
    costos sobre el número de iteraciones
    '''
    m = len(y)
    cost_history = np.zeros(iterations)
    theta_history = np.zeros((iterations,2))
    for it in range(iterations):

        prediction = np.dot(X,theta)

        theta = theta -(1/m)*learning_rate*( X.T.dot((prediction - y)))
        theta_history[it,:] =theta.T
        cost_history[it]  = cal_cost(theta,X,y)

    return theta, cost_history, theta_history
```

Regresión Lineal

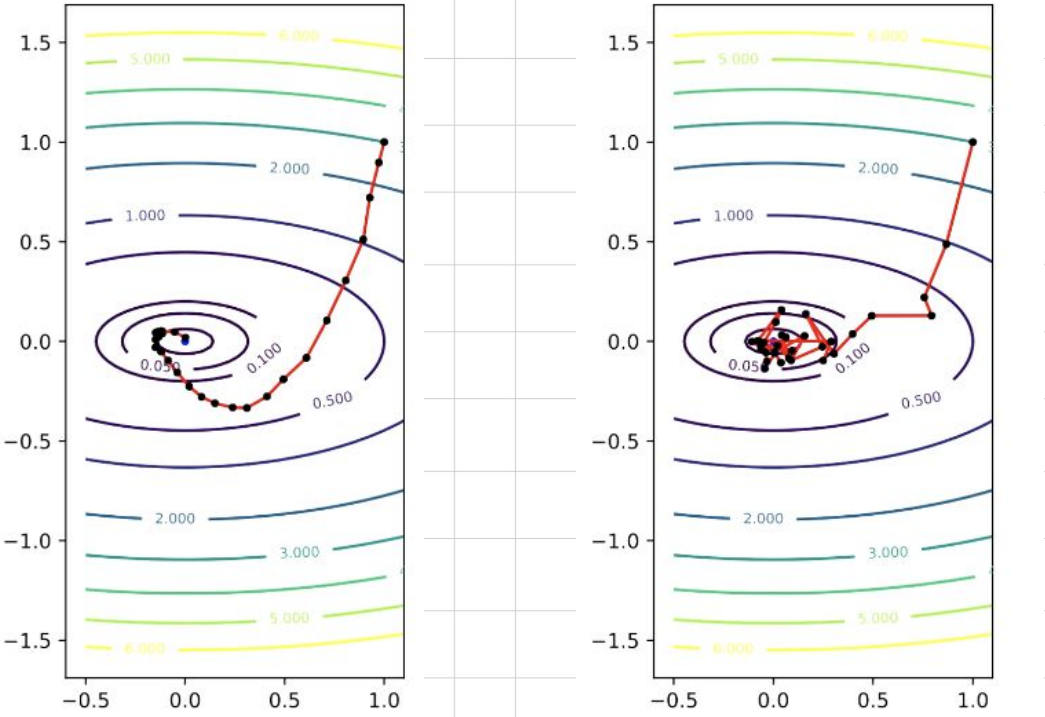


Descenso de gradiente en la función de coste

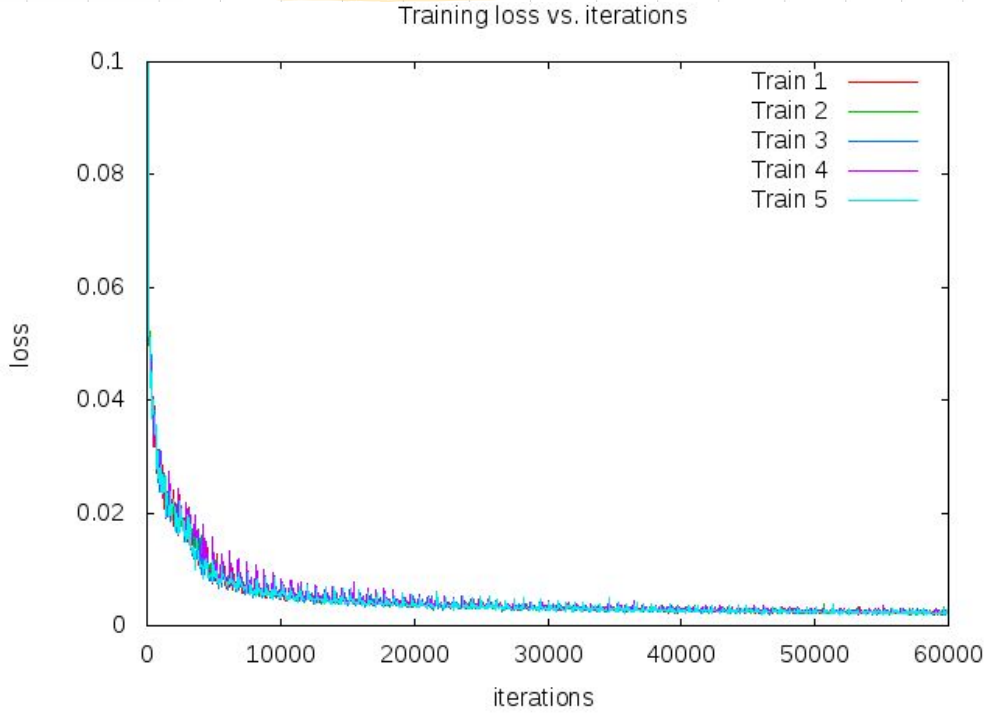




Descenso de gradiente en la función de coste



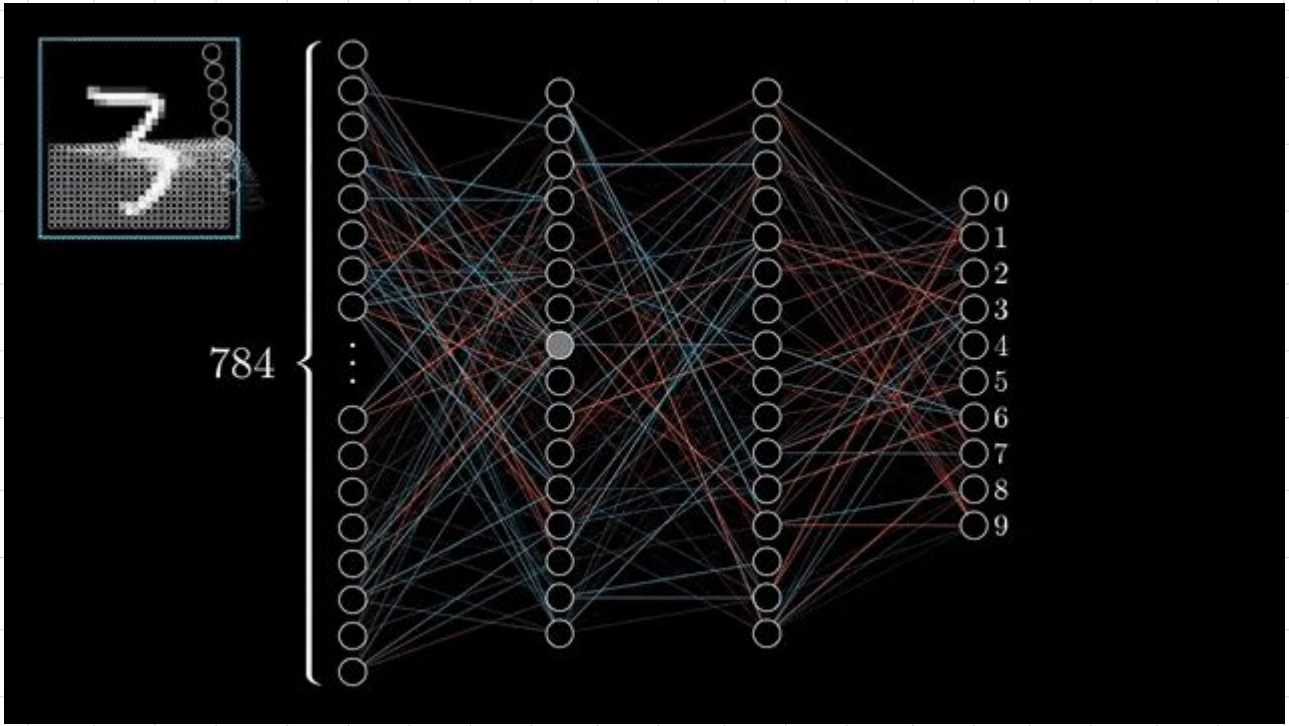
Error de entrenamiento



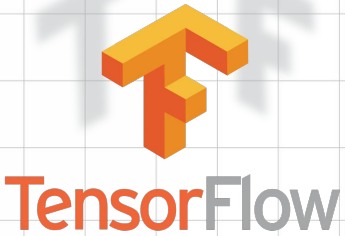
Complejidad

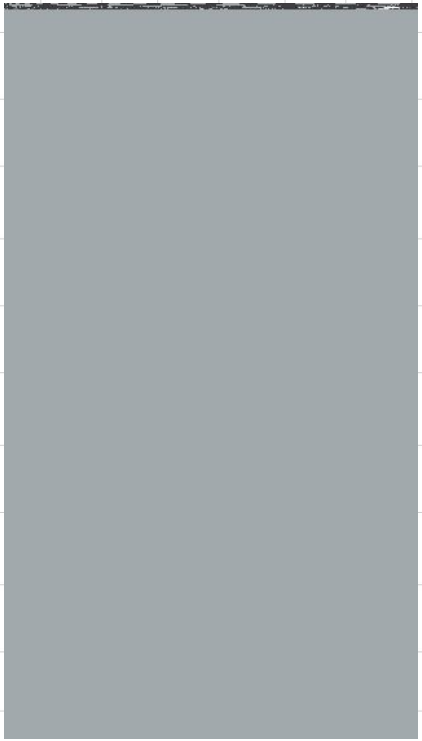
El costo computacional del descenso del gradiente depende del número de iteraciones que se necesitan para converger. La complejidad del descenso por gradiente es $O(kn^2)$, así que cuando N es muy grande se recomienda utilizar el descenso de gradiente en lugar de la forma cerrada de regresión lineal.

Redes Neuronales



Frameworks





Frameworks

PyTorch

Deep Learning - DEMO

Demo UNI