

# Práctica 1: regresión lineal

Andrew Ng

# Gradient descent algorithm

repeat until convergence {

$$\begin{aligned}\theta_0 &:= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) \\ \theta_1 &:= \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) \cdot x^{(i)}\end{aligned}$$

}       $\left. \begin{array}{l} \text{update} \\ \theta_0 \text{ and } \theta_1 \\ \text{simultaneously} \end{array} \right\}$

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import numpy as np
```

```
fig = plt.figure()
ax = fig.gca(projection='3d')
```

```
# Make data.
```

```
X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)
```

```
# Plot the surface.
```

```
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
                      linewidth=0, antialiased=False)
```

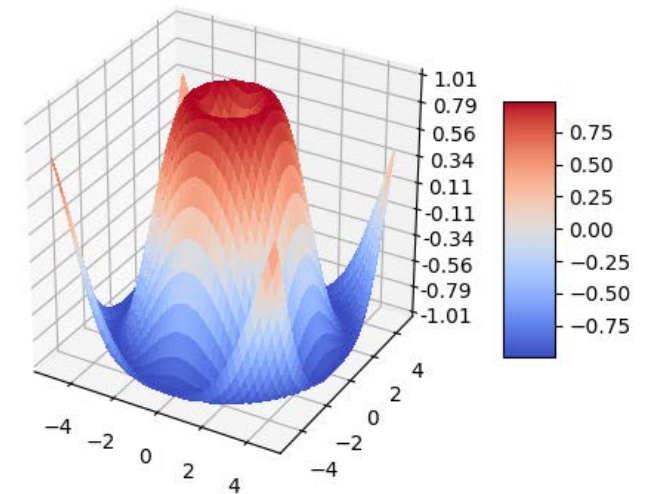
```
# Customize the z axis.
```

```
ax.set_zlim(-1.01, 1.01)
ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))
```

```
# Add a color bar which maps values to colors.
```

```
fig.colorbar(surf, shrink=0.5, aspect=5)
```

```
plt.show()
```

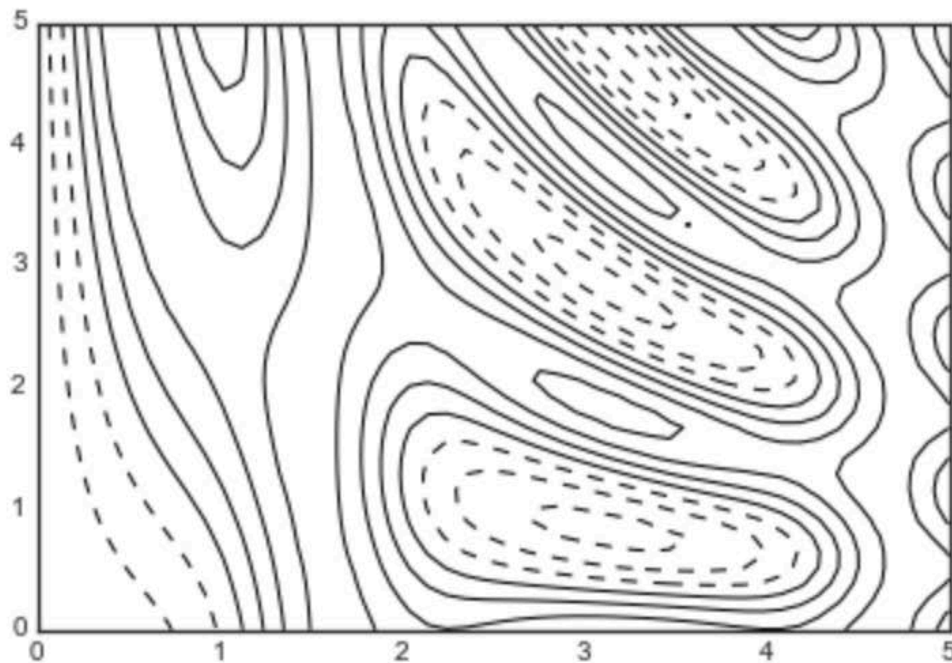


```
def f(x, y):  
    return np.sin(x) ** 10 + np.cos(10 + y * x) * np.cos(x)
```

```
x = np.linspace(0, 5, 50)  
y = np.linspace(0, 5, 40)
```

```
X, Y = np.meshgrid(x, y)  
Z = f(X, Y)
```

```
plt.contour(X, Y, Z, colors='black');
```



```
# El cuarto argumento es una lista con los ticks de las curvas de nivel  
# donde se usa una escala logarítmica de 20 valores entre  $10^{-2}$  y  $10^3$   
CS = ax.contour(Theta0, Theta1, Coste, np.logspace(-2, 3, 20))
```