Actividad 4: Ajuste de datos

Valenzuela Carrillo María Inés

29 de Enero del 2016

1. Introducción

Mínimos cuadrados es una técnica de análisis numérico enmarcada dentro de la optimización matemática, en la que, dados un conjunto de pares ordenados: variable independiente, variable dependiente, y una familia de funciones, se intenta encontrar la función continua, dentro de dicha familia, que mejor se aproxime a los datos, de acuerdo con el criterio de mínimo error cuadrático.

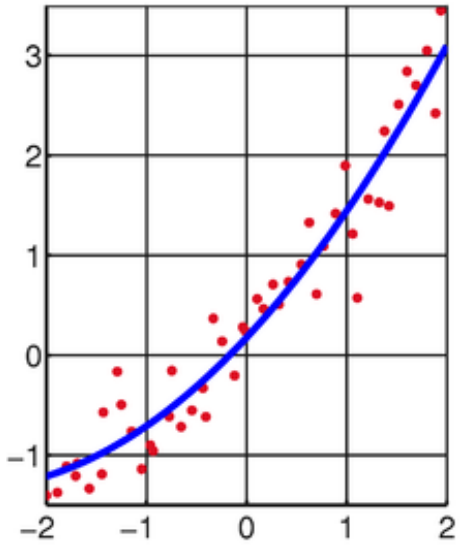


Figura 1: Ejemplo de un ajuste de datos

En su forma más simple, intenta minimizar la suma de cuadrados de las diferencias en las ordenadas (llamadas residuos) entre los puntos generados por la función elegida y los correspondientes valores en los datos. Específicamente, se llama mínimos cuadrados promedio (LMS) cuando el número de datos medidos es 1 y se usa el método de descenso por gradiente para minimizar el residuo cuadrado.[4]

2. Ajuste de datos en Python

En esta actividad se pide realizar ajustes a dos conjuntos de datos dados, temperatura de invierno en Nueva York y presión atmosférica vs. altitud. Para llevar acabo dicho ajuste se utilizaron los paquetes de *Scipy*: *curve_fit; leastsq*.

Scipy obtiene los mejores parámetros para ajustar los valores medidos. La forma de hacerlo es através de una función de error.

2.1. Problema 1: Ajuste Lineal

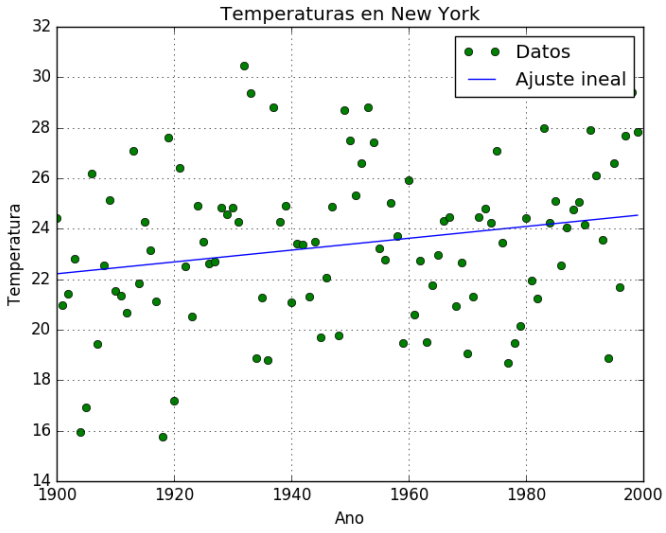
El primer ajuste es para los datos de la temperatura de invierno en New York, en este caso se aplicara una regresión lineal, es decir, la función resultante tendra la forma $y = ax + b$.

El código para la regresión es:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize

#Lectura de datos y separación
datos = np.loadtxt('ny.txt')
x=datos[:,0]
y=datos[:,1]

#Definiendo la forma de la función
fitfunc = lambda p, x: p[0]*x + p[1]
errfunc = lambda p, x, y: fitfunc(p, x) - y
```

```
p0 = [1, 1]

#Ajuste de la recta
p1, success = optimize.leastsq(errfunc, p0[:], args=(x, y))

#Graficando
time = np.linspace(x.min(), x.max(), 100)
plt.plot(x, y, "go", label="Datos")
plt.plot(time, fitfunc(p1, time), "b-", label="Ajuste ineal")

plt.grid()
plt.legend()
plt.title("Temperaturas en New York")
plt.xlabel("Ano")
plt.ylabel("Temperatura")

plt.show()
```

La gráfica resultante es:



En esta gráfica se puede ver que los datos de encuentran muy dispersos y aun asi se puede encontrar la recta de mejor ajuste.

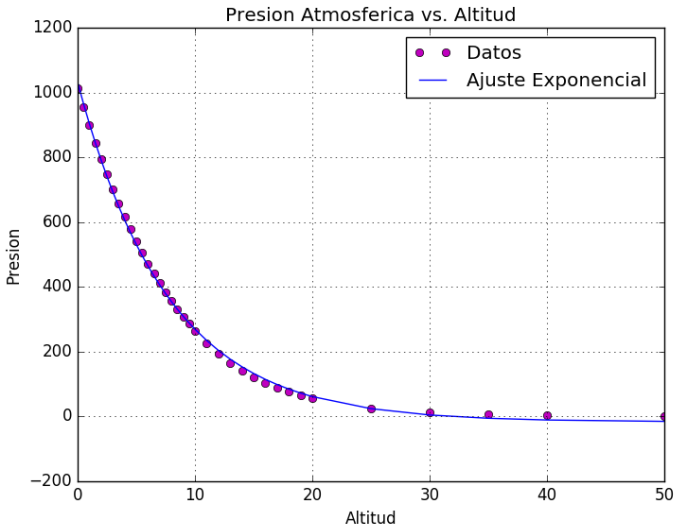
2.2. Problema 2: Ajuste Exponencial

Los datos no siempre se comportaran de manera lineal, para estos casos existen otro tipo de ajustes no lineales. Para los datos de presión contra altitud se ajustó una función exponencial de la forma $ae^{-bx} + c$.

El código para el ajuste exponencial es:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

#Lectura de datos y separación
datos = np.loadtxt('pa.txt')
x=datos[:,0]
y=datos[:,1]

#Definiendo la forma de la función
def f(x, a, b, c):
    return a * np.exp(-b * x) + c


#Ajuste exponencial
popt, pcov = curve_fit(f,x,y)

#Graficando
plt.plot(x, y, "mo", label='Datos')
plt.plot( x, f(x, *popt), "b-", label='Ajuste Exponencial')

plt.grid()
plt.legend()
plt.title("Presion Atmosferica vs. Altitud")
plt.xlabel("Altitud")
plt.ylabel("Presion")

plt.show()
```

La gráfica resultante es:



En la gráfia se puede ver como la función exponencial se ajusta de manera muy precisa a los datos.

Referencias

[1] Wikipedia, the free encyclopedia, *Mínimos cuadrados.*

[2] Temperatura de invierno en New York,
http://www.seattlecentral.edu/qelp/sets/048/048.html.

[3] Presión Atmosférica vs Altitud,
http://www.seattlecentral.edu/qelp/sets/024/024.html.

[4] Actividad 4 (2016-1),
http://computacional1.pbworks.com/w/page/105016164/Actividad4.