Proyecto 3

Ruben Cuadra

Dr. Victor de la Cuva

Aprendizaje Automatico

August 28, 2017

Regresión lineal (recta) Multivariable

Implementación en python de regresión multivariable el cual sucede cuando los datos se aproximan por medio de una hiperrecta (polinomio lineal) dada por la hipótesis :

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots \theta_n x_n = \theta^T X$$

Donde el conjunto de datos X es = $[x_1, x_2, \dots x_n]^t$

El código consta de una librería llamada Proyecto3.py la cual consta de 9 funciones, todas giran entorno a la función: (Las negritas son los parámetros que recibe)

gradienteDescendenteMultivariable:

X : Matrix de valores en X
Y : Vector con valores en Y, mismo tamaño de X
alpha (opcional) : Razón de aprendizaje , por default es 0.01
iteraciones (opcional): Default es 100

Nos devuelve 1 tupla con 2 objetos, el primero es un arreglo con el resultado de la función de costo evaluada en cada iteración, de modo que el arreglo tiene el mismo tamaño que el numero de iteraciones, segundo es el vector theta que posee el mismo tamaño que el numero de variables X + 1

$$\theta_j = \theta_j - \frac{\alpha}{n} \sum_{i=0}^{n} (h(x_i) - y_i) x_i^j$$

*n es la cantidad de datos que tenemos(Renglones en la matriz X o Y)

Ejecuta esa formula *iteraciones* veces y obtiene el costo usando la función *calculaCosto*

gradienteDescendenteMultivariable:

historial : Arreglo de números, nos lo regresa la función *gradiente-DescendenteMultivariable*

Grafica el arreglo

calculaCosto:

X : Matrix de valores en X
Y : Vector con valores en Y
theta : Vector con valores theta

Regresa un valor numérico obtenido de la ecuación

$$J = \frac{2}{n} \sum_{i=0}^{n} (h(x_i) - y_i)^2$$

ecuacionNormal:

X : Matrix de valores en X
Y : Vector con valores en Y

Regresa una matriz con los valores theta usando la ecuación:

$$\theta = (X^T X)^{-1} X^T Y$$

normalizacionDeCaracteristicas:

X : Matrix de valores en X

Regresa una tupla con 3 valores
_X = Matriz normalizada
μ = Vector con la media para cada columna de la matriz X
σ = Vector con desviaciones estándares para cada columna

La normalisation de la matriz se da por la formula

$$x_i = \frac{x_i - \mu}{max(x) - min(x)}$$

predicePrecio:

X : Vector de valores
Theta : Vector de valores en Theta

Regresa un valor numérico que debería corresponder a Y, evalua la hipótesis con esos parámetros

_X = Matriz normalizada
μ = Vector con la media para cada columna de la matriz X
σ = Vector con desviaciones estándares para cada columna

La normalisation de la matriz se da por la formula

$$x_i = \frac{x_i - \mu}{max(x) - min(x)}$$

predicePrecio

Requisitos

Libreria matplotlib (Graficas)

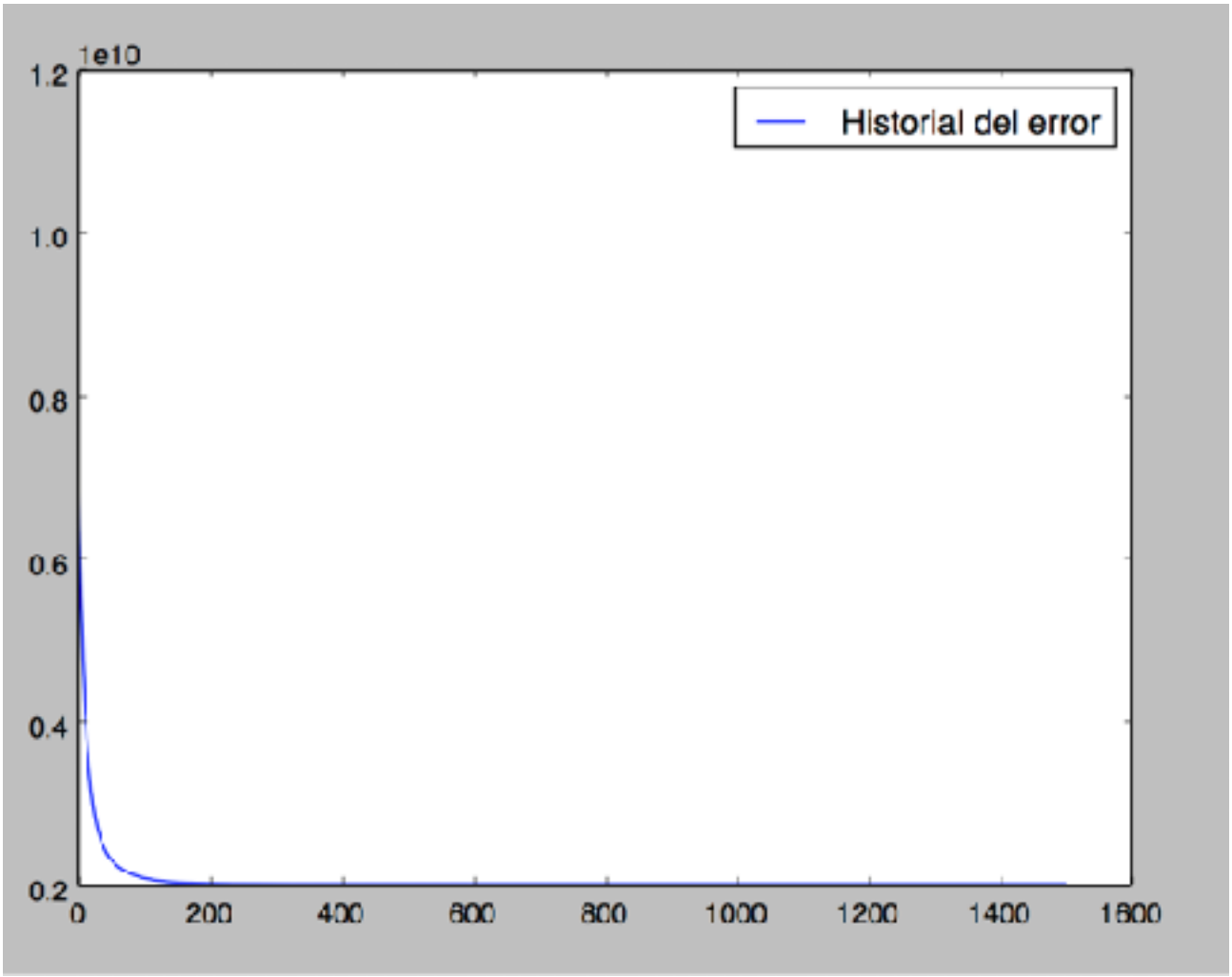
Libreria numpy (Operaciones matemáticas)

Python 2.7 o 3.5 (Instalar las librerías correctamente usando pip o pip3)

En el repositorio se encuentra un archivo ejemplo que es parseado por la funcion *getDataFromFile* la cual recibe como parámetro un archivo(su ruta absoluta) y nos devuelve la matriz X y el vector Y, los cuales ya pueden ser usados para todas las funciones previamente descritas **datos.csv** contiene valores numéricos x,y ; Son N columnas separadas por comas donde la ultima representara las Y y las demás valores en X.

Tras correr una serie de pruebas con la información ejemplo se llego a la conclusión de obtener la información del archivo usando **getDataFromFile** (Ya tendremos X y Y)posteriormente usaremos **normalizacionDeCaracteristicas** con el resultado en X de la función anterior, una vez que obtenemos una X normalizada se manda a llamar la función **gradienteDescendenteMultivariable** con la X normalizada, el vector Y, una **alpha de 0.729** y con *iteraciones* a 1500 y obtenemos esta gráfica de error:

Proyecto 3


1e10
1.2
1.0
0.8
0.6
0.4
0.2
0
200
400
600
800
1000
1200
1400
1600
Historial del error