Proyecto 3

Regresión lineal (recta) Multivariable

Descripción

El caso más simple de la regresión múltiple es cuando la aproximación se hace por medio de un polinomio lineal (hiperrecta) de la forma:

$$f(\boldsymbol{X}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Donde $\boldsymbol{X} = [x_1, x_2, \dots, x_n]^T$.

Hacer un programa en Python que reciba como entrada un conjunto de ejemplos de la forma **(entradas,salida)**, es decir, $(x_1, x_2, \dots, x_n)$, y regrese los valores de Theta ($\theta_0 \dots \theta_n$) que mejor ajusten al conjunto de datos. La recta debe ser calculada por el algoritmo de regresión lineal.

Notas

- Recuerde que para facilitar la vectoriazación es necesario agregar una columna de 1´s a la matriz de entradas X.
- Para poder vectorizar tiene que contar con operaciones con matrices por lo que puede usar una librería que las proporcione.

Datos

Los datos están en un archivo (ex1data2.txt) de 3 columnas (Tamaño en feet^2, Número de cuartos y precio) que contiene los datos de costos de casas en Portland, Oregon. El archivo deberá ser separado en dos vectores:

1. El vector **X** conteniendo TODAS las entradas. Se refieren a la cantidad de feet^2 de la casa y numero de cuartos.
2. El vector **y** conteniendo TODAS las salidas. Se refieren al precio de la casa en USD.

Fuente: A. Ng. Machine Learning. Curso de Coursera-Stanford (2011).

Notas

- Recuerde que para facilitar la vectoriazación es necesario agregar una columna de 1´s a la matriz de entradas X.
- Para poder vectorizar tiene que contar con operaciones con matrices por lo que puede usar una librería que las proporcione.
- Los datos se deben normalizar.
- Seleccione y reporte la tasa de aprendizaje, mediante la prueba descrita en clase, corriendo gradiente descendente alrededor de 50 iteraciones y probando valores de la tasa iniciando con 0.001 y multiplicando por 3 en cada ocasión.

Funciones

Se deberán programar 6 funciones

1. **`normalizacionDeCaracteristicas(X)`**. Recibe el vector de entradas X y regresa tres datos:
    a. **X**. Un vector de entradas normalizadas (sin el vector de 1's).
    b. **mu**. Un vector de las medias de cada una de las características.
    c. **sigma**. Un vector de las desviaciones estándar de cada una de las características.

    Esta función debe poder trabajar con conjuntos de datos de cualquier dimensión (no sólo para el archivo de ejemplos dado).

2. **`gadienteDescendenteMultivariable(X,y,theta,alpha,iteraciones)`**. Recibe:
    a. Datos de entrada ya separados en vectores **X** y **y**.
    b. Vector **theta** = $[\theta_0,\ldots,\theta_n]$ inicial. Dicho vector puede estar inicializado en 0 para todos sus valores.
    c. Razón de aprendizaje **alpha**. Se recomienda probarlo con un valor alpha = 0.1 pero, finalmente, usará la que calculó con el proceso dado.
    d. El número de iteraciones **iteraciones**. El número de iteraciones que va a realizar el algoritmo. Se recomienda probarlo con iteraciones = 100.

    La función deber regresar el valor del vector **theta** final así como un vector del historial del error **J_Historia** (de tamaño igual al número de iteraciones).

3. **`graficaError(J_Historial)`**. Recibe un vector que contiene el historial del error para una corrida del gradiente descendente y lo grafica con respecto al tiempo (número de iteraciones).

4. **`calculaCosto(X,y,theta)`**. Recibe las entradas y un vector theta y debe regresar la función de costo $J(\theta_0,\ldots,\theta_n)$ que resulta.

5. **`ecuacionNormal(X,y)`**. Recibe los vectores **X** y **y** directos del archivo de entrada y regresa el vector **theta** obtenido con la forma directa $\boldsymbol{\theta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$. Aunque no se requiere hacer ninguna normalización sí se debe agregar el vector de 1's a X para hacer la vectorización más sencilla.

6. **`predicePrecio(X,theta)`**. Recibe un vector de entradas X (para un solo ejemplo) y un vector theta. Regresa el precio de la casa que se predice para dicha entrada.

Salve su archivo con el nombre **Proyecto3.py**. Tiene que ser con ese nombre porque es el que se va a usar para la revisión automática.

Resultados de prueba

Una vez entrenado el sistema, es decir, una vez encontrados los valores de los parámetros para la recta de regresión, se pueden hacer las siguientes pruebas:

1. Hacer la predicción con el vector θ encontrado por gradiente descendente normalizando las entradas.
2. Hacer la predicción con el vector θ encontrado por la ecuación normal sin normalizar las entradas.
3. Ambas salidas deben ser muy parecidas.

Recuerde que $\boldsymbol{h_\theta(x) = \theta^T X = \theta_0 + \theta_1 x_1 + \cdots \theta_n x_n}$ donde $\boldsymbol{X}$ representa en este caso el vector de ejemplos completo, es decir, con las parejas ordenadas **(entradas, salida)**.

Pueden también imprimir el costo al inicio, antes de llamar a **gradienteDescendente**, con el vector inicial de **theta** igual a **[0,0,…,0]**. Luego lo imprimen después de correr la función y el vector debe disminuir casi hasta 0.

Vectorización

VECTORICE TODAS las operaciones que pueda vectorizar, utilizando alguna librería para manejo de matrices, esto le facilitará la programación. La función de la ecuación normal es la única obligatoria para hacerla vectorizada.

Revisión

Para la revisión se harán pruebas sobre las funciones solicitadas. Para esto, tanto el archivo que contiene las funciones (el suyo) como el de prueba (el mío), se colocarán en el mismo folder.

En el programa de prueba al inicio hará un: `from file import function`

y luego se utilizará en el script de prueba llamándola como como: `function(<parámetros>)`