Proyecto 4

Regresión Logística

Descripción

Hacer un programa en Python que reciba como entrada un conjunto de ejemplos de la forma **(entradas,salida)** y regrese los valores de **Theta (θ_0 … θ_n)** que logran crear una recta para clasificar los datos correctamente. La recta debe ser calculada por el algoritmo de regresión logística. El objetivo del sistema es poder predecir si un alumno pasará un examen de admisión o no para lo cual se tiene el historial (datos de entrada) de dos exámenes de alumno y el resultado de su admisión.

Datos

Los datos están en un archivo (ex2data1.txt) de 3 columnas (examen 1, examen 2 y admisión). El archivo deberá ser separado en dos vectores:

1. El vector **X** conteniendo TODAS las entradas. Se refieren a la los resultados de los exámenes.
2. El vector **y** conteniendo TODAS las salidas. Se refieren al resultado de su admisión.

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

Funciones

Se deberán programar 5 funciones

1. **graficaDatos(X,y,theta)**. Recibe el vector de entradas X, salidas y, y un vector theta. Grafica los datos en 2D. Los ejes son las dos calificaciones y pone una X si resultó admitido y una O si no fue admitido. También grafica la recta cuyos parámetros son los del vector theta dado.
2. **sigmoidal(z)**. Recibe:
    a. Recuerde que la hipótesis de regresión logística se define como
    $$h_\theta = g(\theta^T x)$$
    Donde la función g es sigmoidal. La función sigmoidal está definida como:
    $$g(z) = \frac{1}{1 + e^{-z}}$$
    b. Esta función recibe un valor de z y regresa su valor sigmoidal.
3. **funcionCosto(theta,X,y)**. Recibe el vector de entrada X, el de salida y, y un vector theta. Debe regresar la función de costo **J** y el gradiente **grad**. Es decir, debe regresar las variables **[J,grad]**.
    a. Recuerde que la función de costo en regresión logística es:
    $$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) - \left(1 - y^{(i)}\right)\log\left(1 - h_\theta\left(x^{(i)}\right)\right)]$$
    b. El gradiente del costo es un vector θ donde el j-ésimo elemento (para j=0,1,…n) está definido como:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

4. **aprende(theta,X,y,iteraciones)**. Recibe el vector de entrada X, el de salida y, un vector theta inicial (el cual puede ser el vector 0) y el número de iteraciones que se correrá el gradiente descendente. Debe regresar el vector de parámetros **theta**, encontrado por gradiente descendente. Si todo es correcto, al mandar a llamar a **funcionCosto** con el vector theta encontrado, el resultado debe ser alrededor de 0.203.
5. **predice(theta,X)**. Recibe un vector **theta** y un vector **X** para varios estudiantes. Regresa el vector **p** de predicción sobre su aceptación utilizando un valor de umbral natural de 0.5, es decir, si da más o igual a 0.5 se acepta (regresa un 1) y si no, se rechaza (regresa 0). Si recibe m estudiantes, regresa un vector **p** de m predicciones. Si todo es correcto, un estudiante con examen 1 de 45 y examen 2 de 85, tendrá una probabilidad de ser admitido de 0.774 y entonces predice 1.

Vectorización y duda

Por favor, VECTORICEN TODAS las operaciones que pueda vectorizar.
¿Es necesario agregar la columna de 1's a X para facilitar la vectorización?

Entrega y Revisión

- Salve su archivo con el nombre **Proyecto4.py**. Tiene que ser con ese nombre porque es el que se va a usar para la revisión automática.
- Suba sus archivos dos archivos, el del proyecto (.py) y el de la documentación (.docx o .pdf) **por separado**, NO los ponga en ZIP.
- 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. 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>)`