Practica03: Regresión Logística.

Continuando con el estudio, vamos a intentar predecir si un juego va a ser bueno o malo, basándonos en la opinión de los usuarios. Para ello vamos a establecer que cualquier juego por debajo del 7 se considera malo y por encima del 7 se considera bueno. Como datos de entrada, vamos a utilizar los mismos que en la práctica 2.

Se pide:

Ejericio 1:

Transformar los datos para que la salida sea 0 o 1 en función de si la nota es por debajo de 7 (sería 0 = malo) o 7 o más (sería 1 = bueno.)

Dibujad usando mathplotlib una única gráfica donde se vea la distribución de las clases.

Ejericio 2:

Implementar la clase **class LogisticRegMulti:** del fichero **LogisticRegMulti.py** en su versión vectorial solamente. Deberá heredar de la versión vectorial de LinearRegMulti que hayáis implementado. Las funciones a implementar son las mismas, pero hay que intentar aprovechar al máximo el código ya implementado en la versión de una variable y varias variables. Se puede modificar la clase LinearReg base y la clase LinearRegMulti, pero debeis aseguraros que de los test de la práctica 1 y la práctica 2 se siguen pasando. Para ello habrá que hacer un correcto uso de la programación orientada a objetos.

Recordemos que la función a optimizar en el caso de la regressión logística es la siguiente:

$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w}\vec{x}+b)}}$$

La función de coste o Loss sería:

$$J(\vec{w}, b) = -\frac{1}{m}\sum_{i=1}^{m}[y_i log(y_i') + (1 - y_i) log(1 - y_i')]$$

Donde

$$y' = f_{\vec{w},b}(\vec{x_i})$$

Y el gradiente sería:

$$w_j = w_j - \alpha\frac{1}{m}\sum_{i=1}^{m} y_i' - y_i x_{j,i}$$

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} y_i' - y_i$$

Ejercicio 3

Incorporad la regularización L2 tanto para el coste como para el gradiente.

Ejercicio 4

Ejecutad los test que están en el fichero practica03.py:

- test_cost(x_train, y_train)
- test_gradient(x_train, y_train)
- test_gradient_descent(x_train, y_train)

Y comprobad que se pasan correctamente.

Opcional:

Convertir los valores de la clase en 5 clases diferentes: 0 : malo, menos que 5. 1 : regular, menos que 7 2 : notable, menos de 9 3 : sobresañoemte, menos de 9.5 4 : Must Have, 9.5 o más.

Y calcular al regresión logística para la multiclase.

English verison

Continuing with the study, we are going to try to predict whether a game is going to be good or bad, based on the users' opinion. To do so, we are going to establish that any game below 7 is considered bad and above 7 is considered good. As input data, we are going to use the same as in practice 2.

We ask:

Exercise 1:

Transform the data so that the output is 0 or 1 depending on whether the score is below 7 (would be 0 = bad) or 7 or above (would be 1 = good).

Draw using mathplotlib a single graph showing the distribution of the classes.

Exercise 2:

Implement the class **class LogisticRegMulti:** from the file **LogisticRegMulti.py** in its vector version only. It must inherit from the vector version of

LinearRegMulti that you have implemented. The functions to implement are the same, but try to make the most of the code already implemented in the single-variable and multi-variable version. You can modify the base LinearReg class and the LinearRegMulti class, but you must make sure that the tests of practice 1 and practice 2 are still passed. For this you will have to make a correct use of object oriented programming. Recall that the function to be optimized in the case of logistic regression is the following:

$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w}\vec{x}+b)}}$$

The cost function or Loss would be:

$$J(\vec{w}, b) = -\frac{1}{m}\sum_{i=1}^{m}[y_i log(y_i') + (1 - y_i) log(1 - y_i')]$$

where

$$y' = f_{\vec{w},b}(\vec{x_i})$$

And the gradient would be:

$$w_j = w_j - \alpha\frac{1}{m}\sum_{i=1}^{m} y_i' - y_i x_{j,i}$$

$$b = b - \alpha\frac{1}{m}\sum_{i=1}^{m} y_i' - y_i$$

Exercise 3

Incorporate L2 regularization for both cost and gradient.

Exercise 4

Execute the tests that are in the file practica03.py: - test_cost(x_train, y_train) - test_gradient(x_train, y_train) - test_gradient_descent(x_train, y_train) And check that they are passed correctly.

Optional:

Convert class values into 5 different classes: 0 : bad, less than 5. 1 : fair, less than 7 2 : remarkable, less than 9 3 : outstanding, less than 9.5 4 : Must Have, 9.5 or more. And calculate the logistic regression for the multiclass.