**REPORTE DE PRÁCTICA**

**IDENTIFICACIÓN DE LA PRÁCTICA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Práctica** | **1** | **Nombre de la práctica** | | **Regresión lineal univariable** |
| **Fecha** | **02/10/2021** | **Nombre del profesor** | | **Alma Nayeli Rodríguez Vázquez** |
| **Nombre del estudiante** | | | **Mariana Ávalos Arce** | |

**OBJETIVO**

|  |
| --- |
| El objetivo de esta práctica consiste en implementar la técnica de regresión lineal univariable en Matlab/Octave y en Python. |

**PROCEDIMIENTO**

|  |
| --- |
| Realiza la implementación siguiendo estas instrucciones. |
| Implementa el método de regresión lineal en Matlab/Octave y en Python. Para ello, considera los siguientes requerimientos:   * Utiliza el set de datos del archivo “dataset\_RegresionLineal.txt”. * Utiliza los siguientes valores para los parámetros iniciales:   a0=0 a1=0 beta=0.023 iteraciones=600   * Reporta el errorJ y el valor final de a0 y a1. Además, reporta el valor de h para el dato de prueba x= 9.7687, cuya salida correcta es y= 7.5435. * Comprueba tus resultados con los siguientes:   J=4.4869 a0=-3.5657 a1=1.1599  Dato de prueba x=9.7687. Salida correcta y= 7.5435. Predicción h=7.7648 |

**IMPLEMENTACIÓN EN MATLAB/OCTAVE**

|  |
| --- |
| Agrega el código de tu implementación en Matlab/Octave en el siguiente espacio. |
| % code for linear regression using training data set file  close all % close windows  clear all % clear variables  %training data  data = load("dataset.txt")  % data is a matrix of 97 rows and 2 cols  x = data(:,1);  y = data(:,2);  figure(1)  plot(x, y, 'ok', 'MarkerFaceColor', 'y', 'MarkerSize', 5)  xlabel("x: characteristics (house area)")  ylabel("y: correct output (house price)")  title("Data Plot")  hold on  % initial parameters  a0 = 0; % 1  a1 = 0; % 1  beta = 0.023; % change this  iterMax = 600; % change this  % size of training data  m = numel(x);  iter = 1;  % hypothesis: vector result called h  h = a0 + a1\*x;  % if you dont plut markers, it will be plotted as line  plot(x, h, 'r');  % sum sums all vector elements  % . for element by element operation  J = (1/(2\*m))\*sum(((h - y).^2));  conv = [];  while(iter <= iterMax)  a0 = a0 - beta\*(1/m)\*sum(h - y);  a1 = a1 - beta\*(1/m)\*sum((h - y).\* x);  h = a0 + a1\*x;  %plot(x, h, 'g')  %pause(1)  J = (1/(2\*m))\*sum(((h - y).^2));  conv(iter) = J; % conv[iter]  iter = iter + 1;  end  figure(1)  plot(x, h, 'g')  figure(2)  plot(conv, 'b')  xlabel("Number of Iterations")  ylabel("Error J")  title("Convergence Plot")  input\_data = 9.7687; % must be normalized as well  output\_h = a0 + a1\*input\_data;  figure(1)  plot(input\_data, output\_h, 'ok', 'MarkerFaceColor', 'm', 'MarkerSize', 8)  fprintf('J = %.4f a0 = %.4f a1 = %.4f \nTest: \nx = %.4f y= 7.5435 h = %.4f \n', J, a0, a1, input\_data, output\_h) |

**IMPLEMENTACIÓN EN PYTHON**

|  |
| --- |
| Agrega el código de tu implementación en Python en el siguiente espacio. |
| import matplotlib.pyplot as plt  import numpy as np  # example training data from file  f = open("dataset.txt", "r")  lines = f.readlines()  x = [float(line.split(',')[0]) for line in lines]  y = [float(line.split(',')[1]) for line in lines]  # example training data  #x = [9, 12, 24, 45, 10.5]  #y = [1200, 1520, 2300, 3400, 1370]  fig = plt.figure()  fig.add\_subplot()  ax1 = plt.gca()  ax1.scatter(x, y, s=100, color='yellow',marker="o", linewidths=1, edgecolor='black')  a0 = 0.0  a1 = 0.0  beta = 0.023  iterMax = 600  m = len(x)  iter = 1  h = [(a0 + a1 \* x\_val) for x\_val in x]  ax1.plot(x, h, color='r')  sums = [(h\_val - y\_val)\*\*2 for h\_val, y\_val in zip(h, y)]  J = (1.0 / 2.0 \* m) \* sum(sums)  conv = [] # convergence vector  while iter <= iterMax:  sums0 = [(h\_val - y\_val) for h\_val, y\_val in zip(h, y)]  a0 = a0 - beta \* (1.0 / m) \* sum(sums0)  sums1 = [((h\_val - y\_val) \* x\_val) for h\_val, y\_val, x\_val in zip(h, y, x)]  a1 = a1 - beta \* (1.0 / m) \* sum(sums1)  h = [(a0 + a1 \* x\_val) for x\_val in x]  sums = [(h\_val - y\_val)\*\*2 for h\_val, y\_val in zip(h, y)]  J = (1.0 / 2.0 \* m) \* sum(sums)  conv.append(J)  iter += 1  ax1.plot(x, h, color='g')  ax1.set\_title('Univariate Linear Regression')  ax1.set\_ylabel("Price")  ax1.set\_xlabel("Area")  fig2 = plt.figure()  fig2.add\_subplot()  ax2 = plt.gca()  # for convergence plot  min = 0.0  max = iterMax \* 1.0  xs = list(np.arange(min, max, max / len(conv)))  ax2.plot(xs, conv, color='orange')  ax2.set\_title('Error Convergence')  ax2.set\_ylabel("Error (J)")  ax2.set\_xlabel("Iterations")  # plot test data and output  input\_data = 9.7687  output\_h = a0 + a1 \* input\_data  ax1.scatter([input\_data], [output\_h], marker='o', color='magenta', s=100, linewidths=1, zorder=100, edgecolor='black')  plt.show()  print(f"\n\n\n J = {J} a0 = {a0} a1 = {a1} \n Test:\n x = {input\_data} y = 7.5435 h = {output\_h}\n") |

**RESULTADOS EN MATLAB/OCTAVE**

|  |  |
| --- | --- |
| Agrega en los espacios indicados las imágenes de los resultados obtenidos en Matlab/Octave. | |
|  |  |
| Gráfica de convergencia generada en Matlab | Gráfica del resultado final en Matlab  (datos de entrenamiento, línea inicial y final) |
|  | |
| Imagen de los resultados de la ventana de comandos de Matlab en la que se despliegan los valores de J, a0, a1, el dato de prueba x con la salida correcta y y su predicción h | |

**RESULTADOS EN PYTHON**

|  |
| --- |
| Agrega en el siguiente espacio la imagen de los resultados obtenidos en Python en el que se desplieguen los valores de J, a0, a1, el dato de prueba x con la salida correcta y y su predicción h. |
|  |

**CONCLUSIONES**

|  |
| --- |
| Escribe tus observaciones y conclusiones. |
| Resultó muy palpable la diferencia entre un lenguaje y otro, sobre todo respecto a la manipulación de vectores. En Matlab resulta más compacto el código que tenga que ver con operaciones vectoriales, como lo es la Regresión Lineal de una variable, en comparación con Python, a pesar de ser un lenguaje ya muy simple. Además, la propagación del error podría reducirse con ayuda de la gráfica de los valores de J comparado con el número de iteraciones, ya que brinda una idea de más o menos cuándo el error se vuelve constante a lo largo del tiempo. |