Universidad Rey Juan Carlos

Práctica 1. Regresión y Clasificación con Modelos Lineales y Logísticos (Matlab)

Gonzalo Vega Pérez

(g.vega.2020@alumnos.urjc.es)

Julia López Augusto

(j.lopeza.2020@alumnos.urjc.es)

09144517Q

03961534Z

Curso 2023/2024

ÍNDICE

Ejercicio 1. Regresión Lineal

En la siguiente tabla se muestran los datos de entrenamiento compuestos por dos características de entrada y una salida.

x_1	x_2	y
1	1	0
2	1	0.69
3	1	1.1
4	1	1.39
5	1	1.61
6	1	1.79
7	1	1.95
8	1	2.08
9	1	2.2
10	1	2.3

Tabla 1. Datos de entrenamiento para regresión lineal.

A continuación, resuelva los siguientes apartados:

1.1. Realice un script en el que se obtengan los pesos (ω_0, ω_1 y ω_2) que forman la ecuación de la recta que se ajusta a la relación entre los datos de entrada y de salida. Para ello utilice funciones de alto nivel de Matlab y adjunte en la memoria el valor de los pesos obtenidos.

Primero, creamos dos matrices, X e Y. La matriz X contiene los datos de las dos características de entrada organizados en una matriz 10x2. Por otra parte, el vector Y contiene los datos de salida.

Una vez creadas las matrices, utilizamos la función fitlm(X,Y) para crear el modelo de regresión lineal. El modelo devuelto por fitlm() contendrá el valor de los pesos.

Obtenemos los siguientes resultados:

$\omega 0 = 0.243$

$\omega 1 = 0.230$

$\omega 2 = 0$

	Estimate	SE	tStat	pValue
(Intercept)	0.24333	0.16371	1.4864	0.18076
x1	0.23048	0.026384	8.7359	5.1753e-05
x2	0	0	NaN	NaN

Fig. 1. Resultados de ω_0, ω_1 y ω_2 respectivamente del ejercicio 1.1.

1.2. Represente en una figura en 3D el conjunto de datos de entrenamiento, así como la recta que mejor se ajusta a los datos de acuerdo con los valores de los pesos obtenidos en el apartado anterior.

Para representar la recta que mejor se ajusta a los datos de entrenamiento, debemos obtener los puntos que la componen. Para ello utilizamos la función predict(modelo, X) que devolverá una matriz con diez puntos que definen la recta que mejor se ajusta al modelo.

Para crear la figura 3D utilizamos la función scatter3() para representar los datos de entrenamiento y la función plot3() para dibujar la recta calculada.

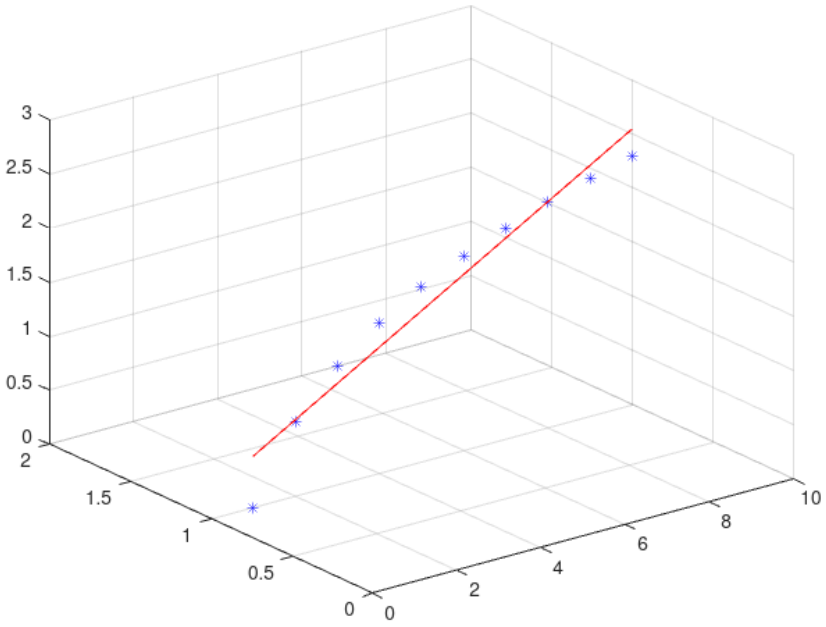

Fig. 2. Resultados visuales del ejercicio 1.2.

```
clc
clear

x1 = [1,2,3,4,5,6,7,8,9,10];
x2 = [1,1,1,1,1,1,1,1,1,1];
X = [x1;x2].';

Y = [0, 0.69, 1.1, 1.39, 1.61, 1.79, 1.95, 2.08, 2.2, 2.3];

% --- 1.1 ---
% Crear modelo de regresión lineal
mdl = fitlm(X,Y);
```

```
%%

% --- 1.2 ---
% Calcular predicciones
ye = predict(mdl, X);

% Mostrar datos iniciales
scatter3(x1, x2 ,Y, 'b*');
hold on
% Mostrar predicción
plot3(x1,x2, ye, 'r');
hold off
```

Fig. 3. Código de los ejercicios 1.1 y 1.2.

1.3. ¿Cree que una recta es la función que mejor se ajusta a la relación entre los datos de entrada y de salida? En caso contrario, indique cuál es la función que mejor se ajustaría.

Analizando la figura 3D del apartado anterior podemos observar que los datos iniciales se distribuyen siguiendo una curva. Es por ello que una función curva como las logarítmicas o las cuadráticas sería más apropiada que una recta.

1.4. Represente la función de coste en función de los parámetros ω_1 y ω_2, suponiendo que ω_0 = 0.24

Para la resolución de este apartado se ha utilizado la Symbolic Toolbox de Matlab.

Haciendo uso de dos bucles *for* podemos programar fácilmente la fórmula matemática que define la función de coste J:

$$J = \frac{1}{2n}\sum_{i=1}^{n}\left(h_\omega\left(x^{(i)}\right) - y^{(i)}\right)^2$$

El primer bucle for se encarga de recorrer cada una de las características de entrada. El segundo bucle permite recorrer cada una de las muestras.

Todos los cálculos los hacemos en función de las variables simbólicas ω_1 y ω_2. Finalmente obtenemos una ecuación simbólica para J que podemos representar utlizando la función fmesh(J)

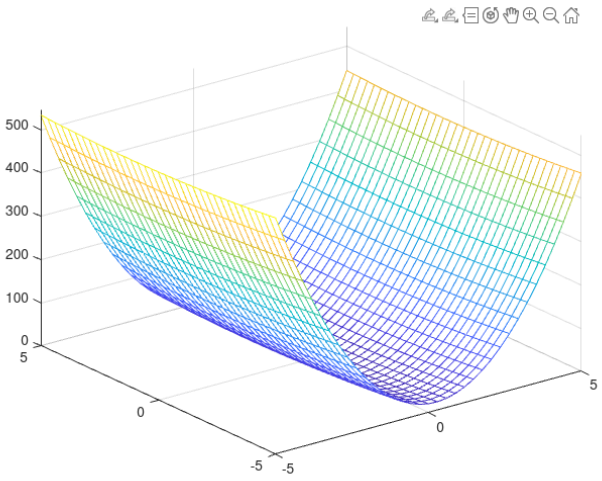

Fig. 4. Resultados visuales del ejercicio 1.4.

```
% --- 1.4 ---
% Obtener función de coste (J)
syms w1 w2;
w0 = 0.24;
%w1 = 0.23;
%w2 = 0;
W = [w1,w2];
n = 10;  % número de muestras
total = 0;
for c = 1:2
    for i = 1:n
        h = w0 + W(c)*X(i, c);
        total = total + (h - Y(i))^2;
    end
end

% Función de coste
J = (1/(2*n)) * total;

fmesh(J)
```

Fig. 5. Código para calcular y representar la función de coste (ej. 1.4).

Ejercicio 2. Regresión Logística Binaria

Se desea construir un modelo basado en regresión logística binaria que sea capaz de clasificar los datos de la tabla adjunta.

x_1	x_2	x_3	y
0.89	0.41	0.69	+
0.41	0.39	0.82	+
0.04	0.61	0.83	0
0.75	0.17	0.29	+
0.15	0.19	0.31	0
0.14	0.09	0.52	+
0.61	0.32	0.33	+
0.25	0.77	0.83	+
0.32	0.23	0.81	+
0.40	0.74	0.56	+
1.26	1.53	1.21	0
1.68	1.05	1.22	0
1.23	1.76	1.33	0
1.46	1.60	1.10	0
1.38	1.86	1.75	+
1.54	1.99	1.75	0
1.99	1.93	1.54	+
1.76	1.41	1.34	0
1.98	1.00	1.83	0
1.23	1.54	1.55	0

Tabla 2. Datos de entrenamiento para regresión logística.

2.1. Realice un script que construya el modelo de clasificador basado en regresión logística, utilizando funciones de alto nivel de Matlab. Adjunte en la memoria los datos del modelo generado que se pueden visualizar en la línea de comandos de Matlab.

Para resolver el ejercicio es necesario crear un array inicializado con cada una de las entradas: X = [x1; x2; x3] y también las salidas con Y.

Para poder crear el modelo de regresión logística binaria, es necesario utilizar la función de Matlab fitlm()

```
mdl =


Generalized linear regression model:
    logit(y) ~ 1 + x1 + x2 + x3
    Distribution = Binomial

Estimated Coefficients:
                   Estimate      SE       tStat      pValue
                   ________    ______    ________    _______

    (Intercept)      1.9146    1.2641     1.5146     0.12987
    x1              -0.27488   1.3662     -0.2012    0.84054
    x2              -0.73401   1.5511    -0.47322    0.63605
    x3              -0.90411   2.2513     -0.4016    0.68798


20 observations, 16 error degrees of freedom
Dispersion: 1
Chi^2-statistic vs. constant model: 4.52, p-value = 0.211
```

Fig. 6. Resultados obtenidos tras la ejecución del ejercicio 2.1

```
clc
clear

x1 = [0.89,0.41,0.04,0.75,0.15,0.14,0.61,0.25, ...
    0.32,0.40,1.26,1.68,1.23,1.46,1.38,1.54,1.99,1.76,1.98,1.23];
x2 = [0.41,0.39,0.61,0.17,0.19,0.09,0.32,0.77, ...
    0.23,0.74,1.53,1.05,1.76,1.60,1.86,1.99,1.93,1.41,1.00,1.54];
x3 = [0.69,0.82,0.83,0.29,0.31,0.52,0.33,0.83, ...
    0.81,0.56,1.21,1.22,1.33,1.10,1.75,1.75,1.54,1.34,1.83,1.55];

X = [x1;x2;x3].';
% + = 1; o = 0
Y = [1,1,0,1,0,1,1,1,1,1,0,0,0,0,1,0,1,0,0,0];

% --- 2.1 ---
% Crear modelo de regresión logística binaria
mdl = fitglm(X,Y,'Distribution','binomial')
```

Fig. 7. Código completo del ejercicio 2.1.

2.2. Represente en una figura las entradas y la clase a la que pertenece cada uno de los datos de entrenamiento, así como la predicción de cada una de las entradas utilizando el modelo creado en el apartado anterior. Nota: utilice distintos marcadores y colores de forma que se puedan distinguir las clases y el tipo de salida (de los datos de entrenamiento o de la predicción a partir del modelo).

Para poder graficar las entradas, la clase a la que pertenecen (círculo o cruz) así como la predicción de cada una de ellas hay que realizar los siguientes pasos:

1. Se crea la ventana de visualización con figure.
2. Se obtienen las predicciones con predict()
3. Se recorren las predicciones y las salidas (ambas tienen la misma dimensión, por lo tanto solo hace falta un bucle for) comprobando si cumplen algunos de los criterios:
    a. Si la salida es 0 y el valor de las predicciones es inferior a 0.5 (se asume que es 0): se trata de que es un círculo acierto y, por lo tanto, pintado de azul.
    b. Si la salida es 0 y el valor de las predicciones es superior a 0.5 (se asume que es 1): se trata de que es un círculo fallo y, por lo tanto, pintado de rojo.
    c. Si la salida es 1 y el valor de las predicciones es superior a 0.5 (se asume que es 1): se trata de que es una cruz acierto y, por lo tanto, pintado de azul.
    d. Si la salida es 1 y el valor de las predicciones es inferior a 0.5 (se asume que es 0): se trata de que es una cruz fallo y, por lo tanto, pintado de rojo.
4. Se añade título y nombre a cada uno de los ejes de la ventana de visualización.

No fue posible incluir una leyenda que sólo mostrase un marcador y nombre de cada tipo.

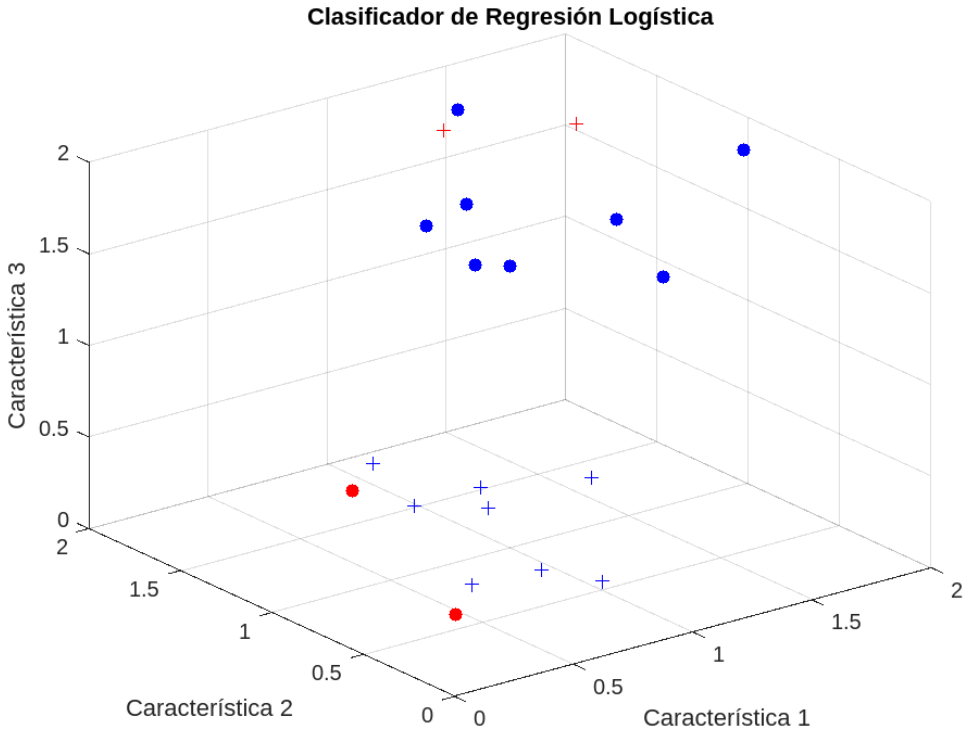

Fig. 8. Imagen obtenida de la ejecución del ejercicio 2.2.

```
% --- 2.2 ---
% representa las entradas
% Crear una figura para visualizar los datos y las predicciones

% Visualización de datos y predicciones
figure;
predictions = predict(mdl, X);

for id = 1:length(predictions)
    if Y(1,id) == 0 && predictions(id,1) < 0.5
        % circulo acierto
        scatter3(X(id, 1), X(id, 2), X(id, 3), 'o', 'filled', 'MarkerFaceColor', 'b');
    elseif Y(1,id) == 0 && predictions(id,1) > 0.5
        % circulo fallo
        scatter3(X(id, 1), X(id, 2), X(id, 3), 'o', 'filled', 'MarkerFaceColor', 'r');
    elseif Y(1,id) == 1 && predictions(id,1) > 0.5
        % cruz acierto
        scatter3(X(id, 1), X(id, 2), X(id, 3), '+', 'MarkerEdgeColor', 'b');
    elseif Y(1,id) == 1 && predictions(id,1) < 0.5
        % cruz fallo
        scatter3(X(id, 1), X(id, 2), X(id, 3), '+', 'MarkerEdgeColor', 'r');
    end
    hold on;
end


% Ajustes de la figura
title('Clasificador de Regresión Logistica');
xlabel('Caracteristica 1');
ylabel('Caracteristica 2');
zlabel('Caracteristica 3');

grid on;
```

Fig. 9. Código para calcular y representar la función de coste (ej. 1.4).

2.3. ¿Cuál es el error en porcentaje que presenta el modelo sobre los datos de entrenamiento? ¿A qué puede ser debido?

Para obtener el error, es necesario de nuevo tomar las predicciones y redondearlas. Seguidamente, crear un contador que vaya recorriendo cada predicción y salida (ambas tienen la misma longitud por lo tanto solo hace falta usar un bucle for) y que vaya aumentando en función de si la predicción y la salida coinciden; si coinciden significará que se trata de acierto.

Después para calcular los errores, es necesario tomar el valor total de muestras y restarle los aciertos. En este caso, hay 20 muestras y 16 de ellas son aciertos; lo que significa de 4 de cada 20 muestras son errores y si finalmente si se multiplica por 100 para poder encontrar el porcentaje: un 20% de error presenta el modelo.

```
62
63    % Calcular el error en porcentaje sobre los datos de entrenamiento
64    train_predictions = predict(mdl, X);
65
66
67    % Convertir las predicciones en etiquetas binarias (0 o 1)
68    predicted_classes = round(train_predictions);
69
70    % Calcular el número de predicciones que coinciden
71    eq = 0;
72    for i = 1:length(predicted_classes)
73       if Y(i) == predicted_classes(i)
74           eq = eq + 1;
75       end
76    end
77
78
79    %Calcular el error obtenido
80    err = ((length(predicted_classes) - eq) / length(predicted_classes)) *100;
81
82    disp("El porcentaje de error de los datos de entrenamiento es: " +err+ "%")
83
84
85
Command Window
New to MATLAB? See resources for Getting Started.
El porcentaje de error de los datos de entrenamiento es: 20%
>>
```

Fig. 10. Código completo y salida obtenida tras ejecutar el ejercicio 2.3.