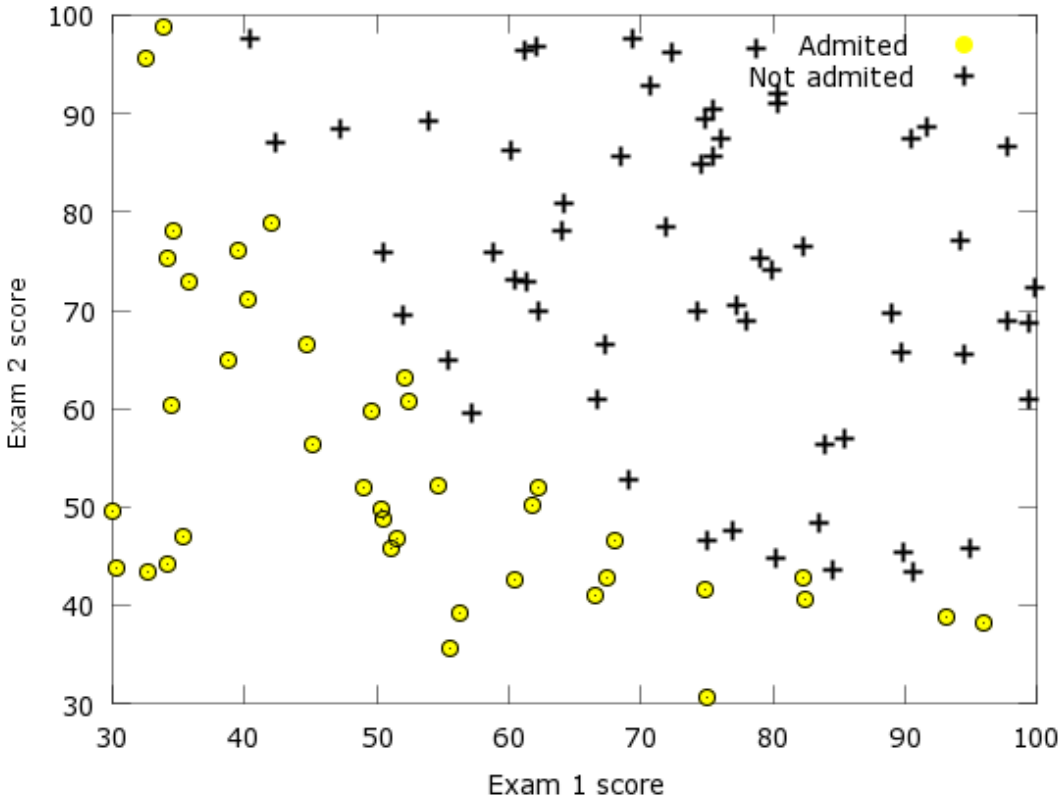
Práctica 2

Ignacio Ferreras y Raúl Martín Guadaño

Parte 1: Regresión logística

En esta primera parte utilizamos el fichero "ex2data1.txt" para realizar un modelo de regresión logística sobre la posibilidad de los estudiantes de acceso a la universidad en función a su nota en dos exámenes.
El fichero contiene esta información en tres columnas: la primera con la nota de un examen, la segunda con la nota del otro examen y la tercera con un 1 o un 0 indicando si fueron admitidos o no respectivamente.

Con los datos del fichero, se pueden ver en función de la nota de los dos exámenes, los alumnos que accedieron a la universidad y los que no representados en esta gráfica:



Realizada con el siguiente código:

```
data = load('ex2data1.txt');
X = data(:, [1,2]);
Y = data(:, 3);
negativos = find ( Y==0) ;
figure;
hold on;
plot (X( negativos , 1) , X( negativos , 2) , 'ko' , 'MarkerFaceColor' , 'y' , ...
'MarkerSize' , 4) ;
positivos = find ( Y==1) ;
plot (X( positivos , 1) , X( positivos , 2) , 'k+' ,  'LineWidth', 2, ...
'MarkerSize' , 4) ;
xlabel('Exam 1 score')
ylabel('Exam 2 score')
legend('Admited', 'Not admited')
hold off;
```

A continuación, implementamos una función *sigmoid* que se corresponde a

$$g(z) = \frac{1}{1+e^{-z}}$$

con el código:

```
function [G] = sigmoid (z)
%func sigmoide
G = 1 ./ (1+ exp(-z));
end
```

Que se puede emplear tanto para un valor como para un vector o una matriz.

Ahora realizamos la función *coste* que devuelve el coste y los valores del gradiente dado un vector theta y los ejemplos proporcionados guardados en X e Y. El código se corresponde con las funciones

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

para el coste, y

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

para el gradiente.

La llamada a la funcion es:

```
[m, n] = size(X);
X = [ones(m, 1), X];
theta = zeros(n+1, 1);
[J, grad] = Coste(theta, X, Y)
```

Y la función *coste* queda así:

```
function [J, grad] = Coste (theta, X, Y)
m = length(Y);
h_theta = 1 ./ (1+ exp(-(X*theta)));
J = 1/m * sum (-Y .*log(h_theta) .- (1-Y).*log(1-h_theta));
grad = 1/m .* (X' * (h_theta .- Y));

end
```

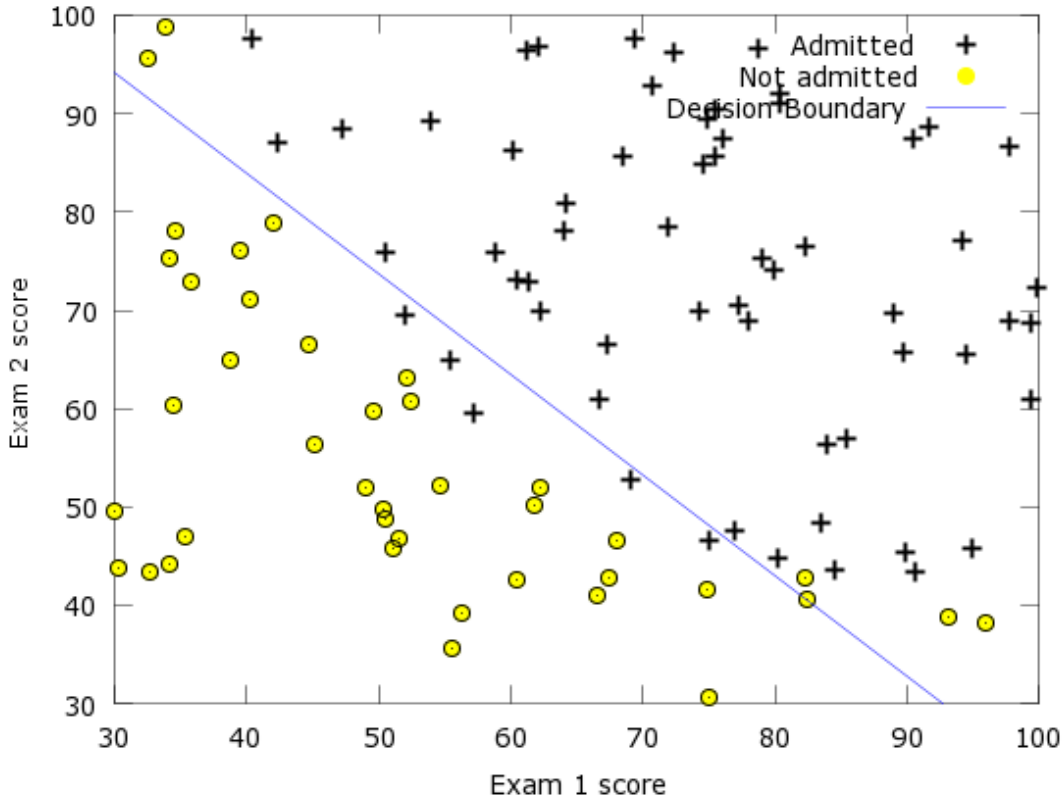
El valor obtenido para la función de coste es 0.69315 y para el gradiente -0.1, -12.00922, -11.26284 respectivamente para cada theta.

Utilizando la función *fminunc* de octave, obtenemos los parámetros theta que hacen mínima la función de coste implementada anteriormente.

La llamada a la función es:

```
opciones = optimset('GradObj', 'on', 'MaxIter', 400);
[theta, J] = fminunc(@(t)(Coste(t,X,Y)), theta, opciones);
plotDecisionBoundary(theta, X, Y);
```

Y como resultado da J=0.20350, theta = -25.16127, 0.20623, 0.20147 y la gráfica:



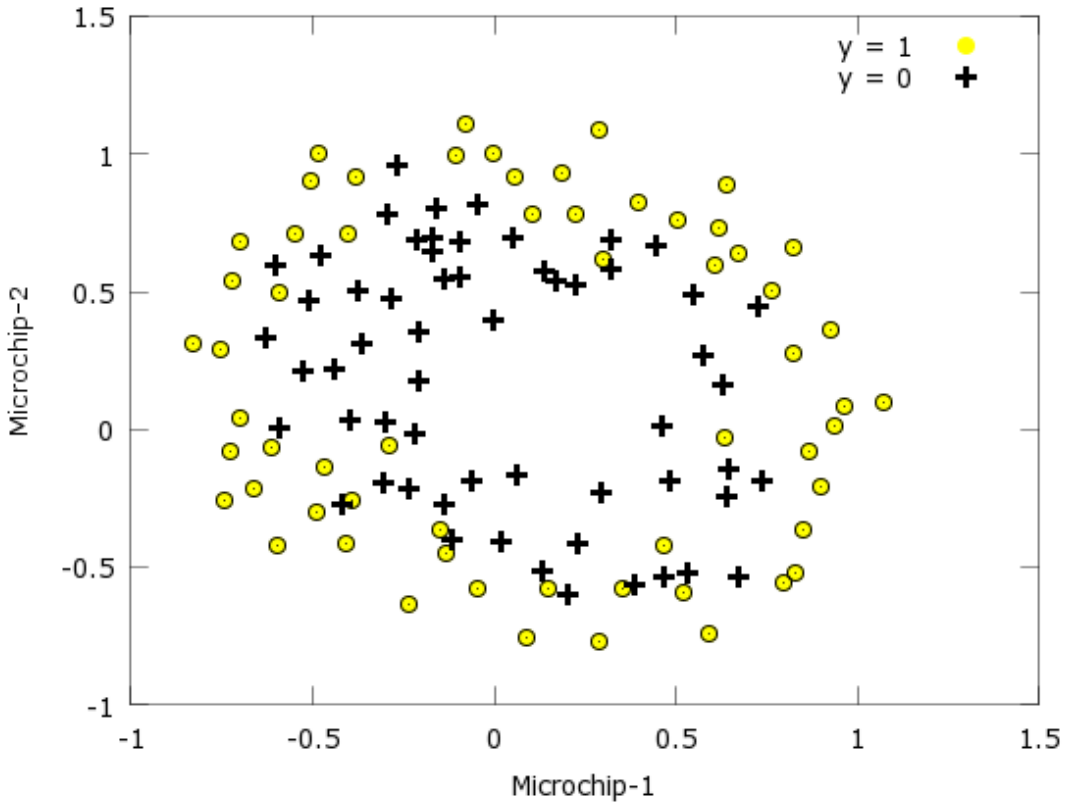
Por último, utilizamos la función *sigmoid* y los valores de theta obtenidos para estimar si se admitirá a cada alumno de los ejemplos de entrenamiento

(>=0.5 se admite). Como resultado obtuvimos un 89% de predicciones correctas. El código es:

```
ej = sigmoid(X*theta);
estimacion = (ej>=0.5);
bien = sum(estimacion==Y)*100/length(Y)
```

Parte 2: Regresión logística regularizada

En este apartado, utilizamos la regresión logística regularizada sobre el fichero "ex2data2.txt" para predecir si un microchip pasará o no el control de calidad. Los datos representados gráficamente son:



Mediante el código:

```
data = load('ex2data2.txt');
X = data(:, [1, 2]);
Y = data(:, 3);
positivos = find(Y == 1);
negativos = find(Y == 0);
figure;
hold on;
plot (X( negativos , 1) , X( negativos , 2) , 'ko' , 'MarkerFaceColor' , 'y' , ...
'MarkerSize' , 4) ;
plot (X( positivos , 1) , X( positivos , 2) , 'k+' ,  'LineWidth', 3, ...
'MarkerSize' , 4) ;
%lo ponemos bonito
xlabel('Microchip-1')
ylabel('Microchip-2')
legend('y = 1', 'y = 0')
hold off;
```

A continuación añadimos nuevos atributos a la descripción de los ejemplos y los combinamos con los originales para hacer un mapeo. Utilizamos la función *mapFeature* proporcionada con la práctica para mapear los atributos. El código que llama a esta función es:

```
X = mapFeature(X(:,1), X(:,2));
```

Con los nuevos atributos obtenidos, calculamos la función de coste y su gradiente de la versión regularizada de la regresión logística:

$$J(\theta) = \left[\frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

$$\begin{aligned}
\frac{\partial J(\theta)}{\partial \theta_0} &= \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} && \text{para } j = 0\\
\frac{\partial J(\theta)}{\partial \theta_j} &= \left(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j && \text{para } j \geq 1
\end{aligned}$$

La llamada a la función es:

```
lambda = 1;
theta = zeros(size(X, 2), 1);
[J, grad] = coste_reg(theta, X, Y, lambda)
```

Y el código de la función:

```
function [J, grad] = coste_reg(theta, X, Y, lambda)
m = length(Y);
n = size(theta);
grad = zeros(n);
h_theta = 1 ./ (1+ exp(-(X*theta)));
theta(1) = 0;
J = (1/m)*sum(-Y .*log(h_theta) .- (1.-Y).*log(1.-h_theta)) + ((lambda/(2*m))*sum(theta.^2));
grad = (1/m)* (X'*(h_theta .- Y) + lambda*theta);

end
```

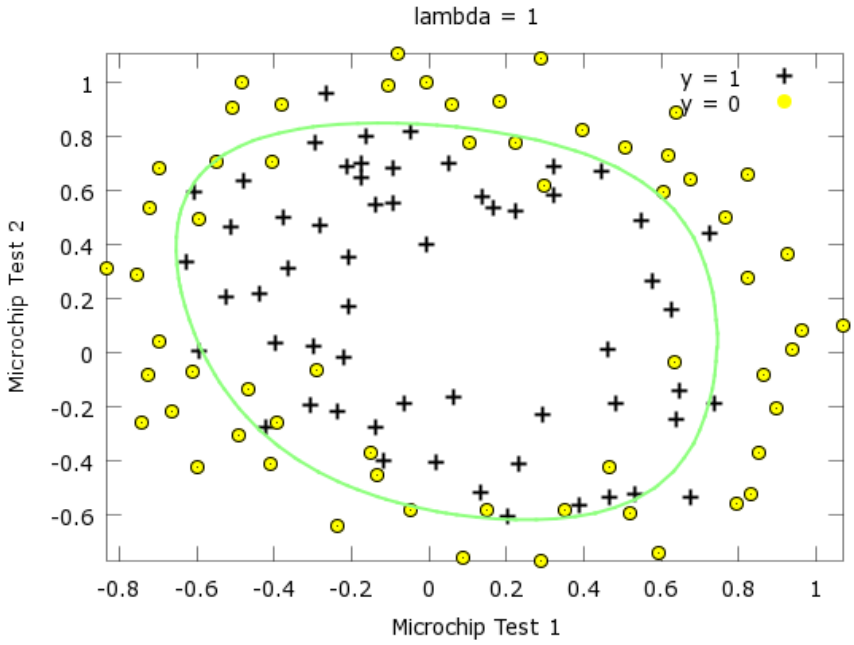
Con este código, los resultados obtenidos fueron:
-Para el coste: 0.69315
-Para el gradiente:

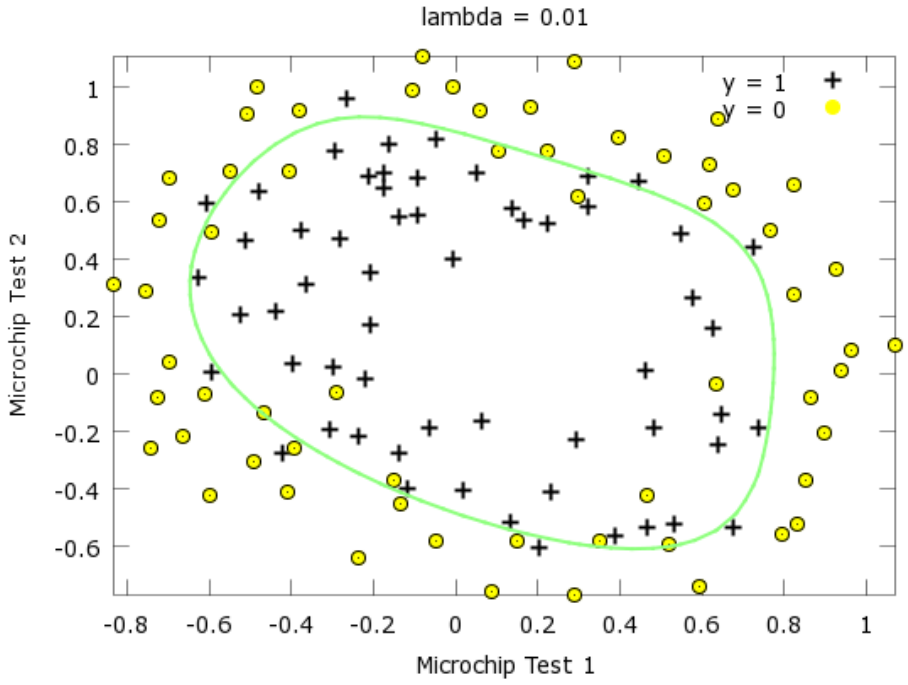
Ahora, empleando la función *fminunc* como en la parte 1 y variando el parámetro lambda se puede observar como varía la gráfica.

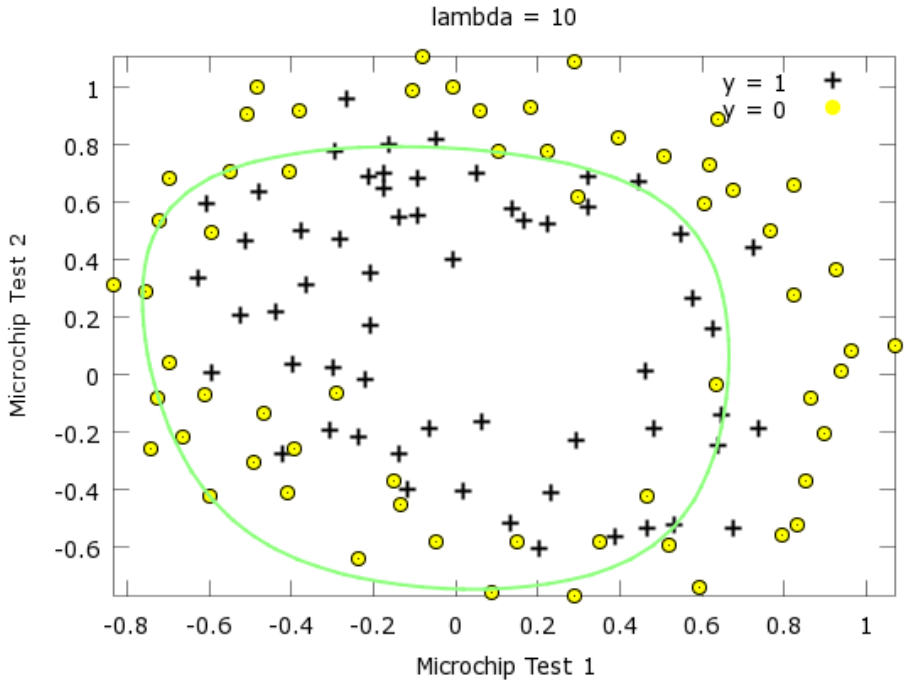
El código empleado es:

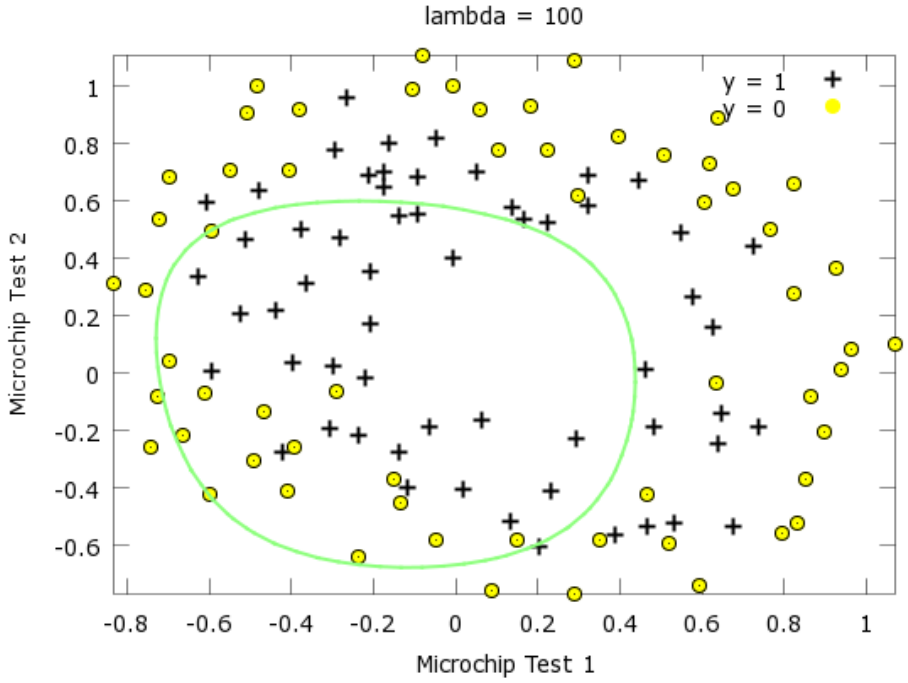
```
theta = zeros(size(X, 2), 1);
lambda = 1;
opciones = optimset('GradObj', 'on', 'MaxIter', 400);
[theta, J] = fminunc(@(t)(coste_reg(t,X,Y, lambda)), theta, opciones);
plotDecisionBoundary(theta, X, Y);
hold on;
title(sprintf('lambda = %g', lambda))
% Labels and Legend
xlabel('Microchip Test 1')
ylabel('Microchip Test 2')
legend('y = 1', 'y = 0', 'Decision boundary')
hold off;
```

Y las gráficas variando las lambdas:


lambda = 1
y = 1
y = 0
Microchip Test 2
Microchip Test 1
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


lambda = 0.01
y = 1
y = 0
Microchip Test 2
Microchip Test 1
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1





Se puede observar como a menor lambda, se agrupan mejor los conjuntos, y a mayor lambda va empeorando el resultado,