Práctica 2: Regresión logística

Fecha de entrega: 22 de marzo de 2018, 16.00h

Material proporcionado:

Fichero	Explicación
`ex2data1.txt`	Datos para la primera parte de la práctica.
`ex2data2.txt`	Datos para la segunda parte de la práctica.
`plotDecisionBoundary.m`	Implementación de la función *plotDecisionBoundary* que pinta el límite de decisión de un conjunto de ejemplos de entrenamientos etiquetados dado por un vector de pesos.
`mapFeature.m`	Implementación de la función que extiende la representación de los ejemplos de entrenamiento con atributos adicionales combinando los atributos originales.

1. Regresión logística

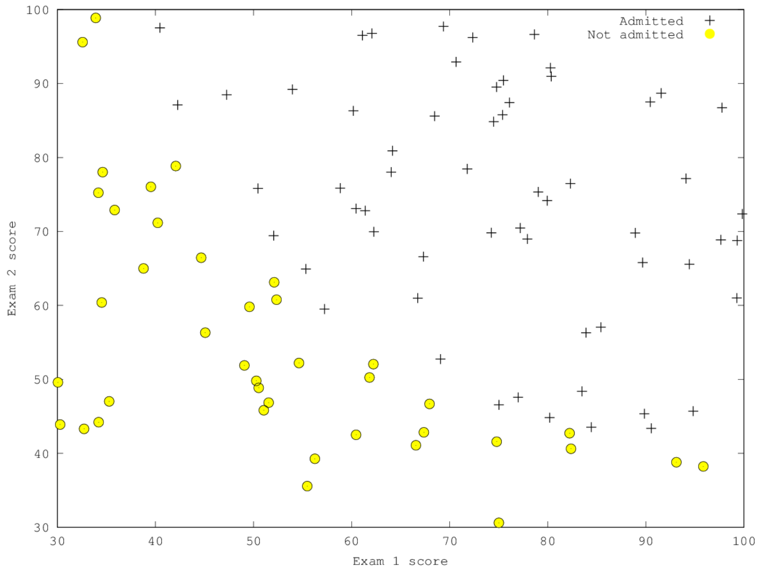
Los datos del fichero `ex2data1.txt` representan las notas obtenidas por una serie de candidatos en los dos exámenes de admisión de una universidad junto con la información sobre si fueron (1) o no (0) admitidos. El objetivo de la práctica es construir un modelo por regresión logística que estime la probabilidad de que un estudiante sea admitido en esa universidad en base a las notas de sus exámenes.

1.1. Visualización de los datos

Para empezar, visualiza el contenido del fichero `ex2data1.txt` en una gráfica similar a la de la figura ayudado por este fragmento de código que visualiza los ejemplos negativos:

```
% Obtiene un vector con los índices de los ejemplos negativos
negativos = find(y==0);

% Dibuja los ejemplos negativos
plot(X(negativos, 1), X(negativos, 2), 'ko', 'MarkerFaceColor', 'y', ...
      'MarkerSize', 7);
```



1.2. Función sigmoide

Implementa una función que calcule el valor de la función sigmoide

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

y que se pueda aplicar indistintamente a un número, un vector o una matriz. En caso de aplicarse a un vector o una matriz, devolverá el resultado de aplicar la función sigmoide a cada uno de sus elementos.

1.3. Cálculo de la función de coste y su gradiente

Implementa una función *coste* que devuelva el valor de la función de coste y un vector con los valores del gradiente de la misma función para un vector de parámetros *theta*, un conjunto de ejemplos de entrenamiento dados en la matriz X y etiquetados con los valores del vector y:

```
function [J, grad] = coste(theta, X, y)
```

Recuerda que el valor de la función de coste en regresión logística viene dado por la expresión:

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

y el gradiente de la función de coste es un vector de la misma longitud que θ donde la componente j (para $j = 0, 1, \ldots, n$) viene dado por la expresión:

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

Inicializando a 0 todos los elementos de *theta* deberías obtener un valor para la función de coste de 0,693, aproximadamente.

1.4. Cálculo del valor óptimo de los parámetros con **`fminunc`**

Debes utilizar la función `fminunc` de Octave para obtener el valor de los parámetros θ que minimizan la función de coste para la regresión logística que has implementado en el apartado anterior. La función `fminunc` recibe tres parámetros:

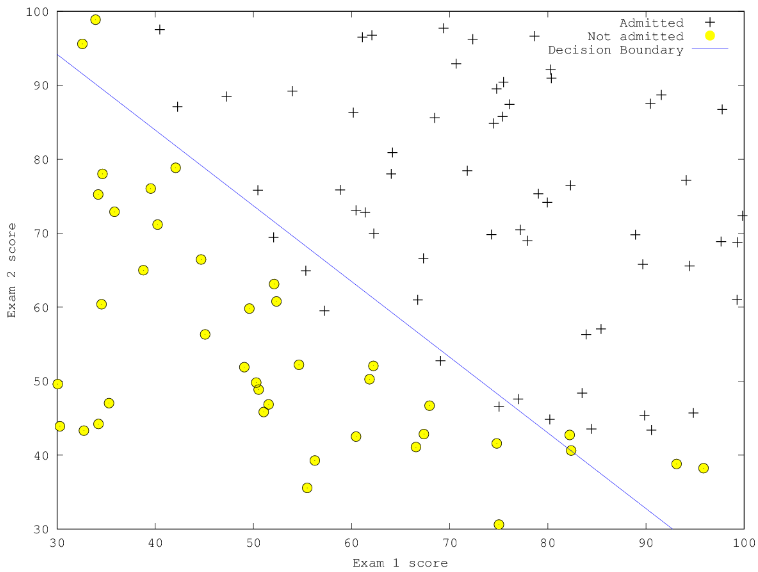
- La referencia a una función que recibe como parámetro el vector de pesos a optimizar y devuelve el coste de la función a minimizar y un vector con su gradiente. Eso es precisamente lo que hace la función *coste* aunque con la salvedad de que recibe como parámetro también una matriz con los ejemplos de entrenamiento y un vector con las etiquetas de esos ejemplos. Por ello, debemos identificar cuál de los tres parámetros de *coste* corresponde a los pesos a optimizar utilizando la sintaxis: `@(t)(coste(t, X, y))`
- Un vector con los valores iniciales de los pesos a optimizar
- Un parámetro con una serie de pares atributo-valor que sirven para configurar a la función `fminunc`.

En resumen, este es el fragmento de código que debes utilizar para invocar a la función de optimización:

```
opciones = optimset('GradObj', 'on', 'MaxIter', 400);

% Obtención del valor óptimo de theta
[theta, cost] = fminunc(@(t)(coste(t, X, y)), theta_inicial, opciones);
```

Si todo va bien, deberías obtener un coste óptimo de aproximadamente 0,203, e invocando a la función `plotDecisionBoundary` con los valores de θ que has obtenido, deberías generar una gráfica similar a esta:



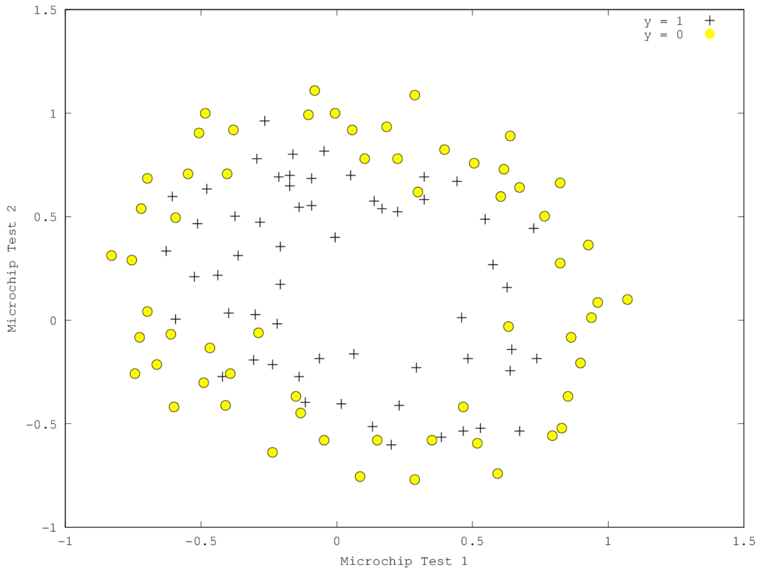
1.5. Evaluación de la regresión logística

En este apartado debes implementar una función que calcule el porcentaje de ejemplos de entrenamiento que se clasifican correctamente utilizando el vector θ que has obtenido en el apartado anterior para calcular el valor de la función sigmoide sobre cada ejemplo de entrenamiento, e interpretando que si el resultado es $\geq 0{,}5$ entonces el alumno será admitido (1) y si es menor no lo será (0).

2. Regresión logística regularizada

En este apartado utilizarás la regresión logística regularizada para encontrar una función que pueda predecir si un microchip pasará o no el control de calidad, a partir del resultado de dos tests a los que se somete a los microchips.

Empieza visualizando los datos, para observar que no son linealmente separables:



2.1. Mapeo de los atributos

Una forma de obtener un mejor ajuste a los ejemplos de entrenamiento usando el método de regresión logística es añadir nuevos atributos a la descripción de los ejemplos, combinando los atributos originales. Utiliza la función *mapFeature* que se proporciona con la práctica para extender cada ejemplo de entrenamiento con los términos polinómicos de x_1 y x_2 hasta la sexta potencia, completando así un total de 28 atributos para cada ejemplo:

$$mapFeature(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \\ x_1^3 \\ \vdots \\ x_2^6 \end{bmatrix}$$

2.2. Cálculo de la función de coste y su gradiente

Implementa una función que devuelva el valor de la función de coste y un vector con los valores del gradiente de la misma función para la versión regularizada de la regresión logística:

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

El gradiente de la función de coste es un vector de la misma longitud que θ donde la componente j se calcula como:

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \qquad \text{para } j = 0$$

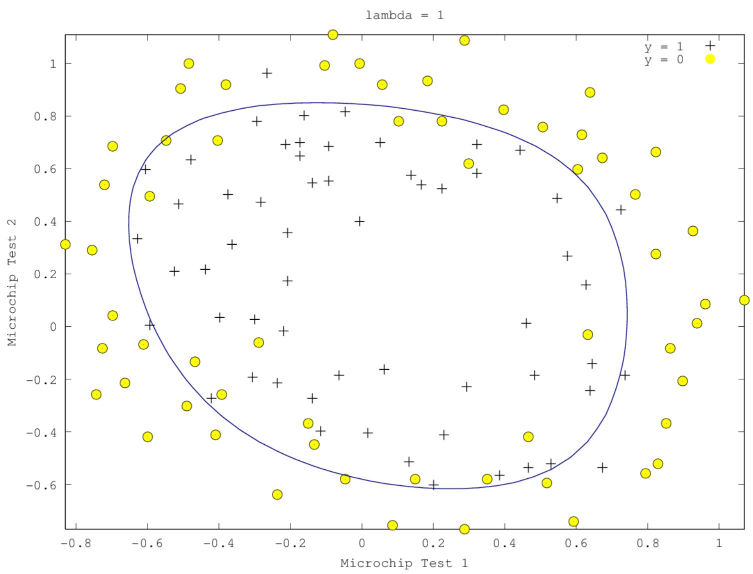
$$\frac{\partial J(\theta)}{\partial \theta_j} = \left(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j \qquad \text{para } j \geq 1$$

Inicializando el vector θ con ceros el coste inicial debería ser de 0,693 aproximadamente.

2.3. Cálculo del valor óptimo de los parámetros con `fminunc`

Utiliza la función `fminunc` para obtener el valor óptimo de θ para la versión regularizada de la función de coste.

El resultado debería ser similar al que se muestra en el figura:



2.4. Efectos de la regularización

Experimenta con distintos valores del parámetro λ para ver cómo afecta el término de regularización al aprendizaje logístico, comparando las gráficas resultantes y evaluando el resultado del aprendizaje sobre los ejemplos de entrenamiento.

3. Entrega de la práctica

La práctica debe entregarse utilizando el mecanismo de entregas del campus virtual, no más tarde de la fecha y hora indicadas en la cabecera de la práctica.

Se entregará un único fichero en formato pdf que contenga la memoria de la práctica, incluyendo el código desarrollado y los comentarios y gráficas que se estimen más adecuados para explicar los resultados obtenidos.