**Práctica 5**

**Pablo Arranz Ropero**

**Juan Alberto Camino Sáez**

**Grupo 2**

**Práctica 5: Regresión lineal regularizada: sesgo y varianza**

Descripción

En primer lugar ejecutamos la siguiente instrucción para cargar los datos necesarios de esta práctica.

1. load(“ex5data.mat”);

Después hemos implementado la función coste en un archivo llamado *coste.m*, que se encarga de calcular el coste y el gradiente de forma vectorizada. Este es el código:

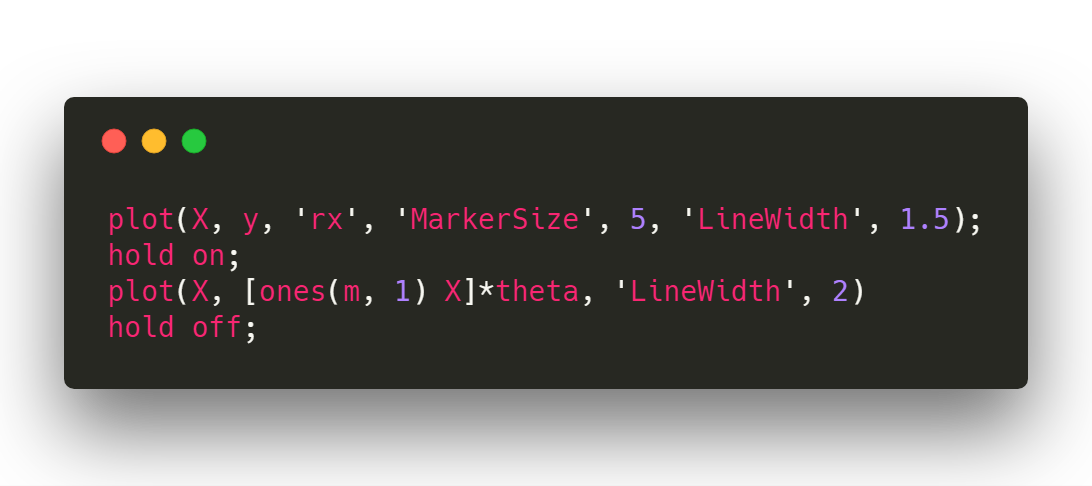


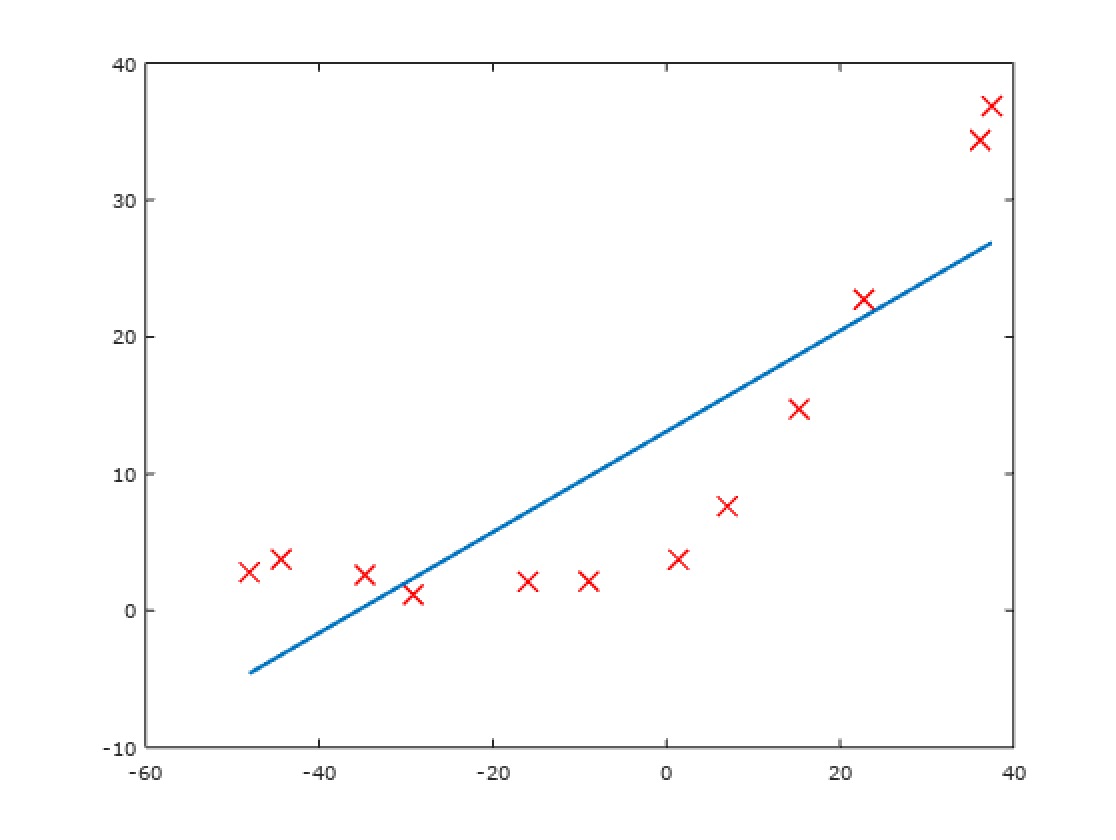
Al comprobar los datos con = [1; 1], dan los datos correctos según indica la práctica, siendo aproximadamente J = 303.99 y el gradiente [−15,30; 598,250].

A continuación, llamamos a la función *fmincg* proporcionada por la práctica, donde le pasamos como argumento esta función, y con 200 iteraciones de la siguiente forma:

theta = fmincg(@(t)(coste(t,X, y, 0)),[1;1],optimset('GradObj', 'on', 'MaxIter', 200));

Al terminar, las thetas resultantes son 0 = 13.087 y 1 = 0.367. Para generar la gráfica, hemos usado el siguiente código:

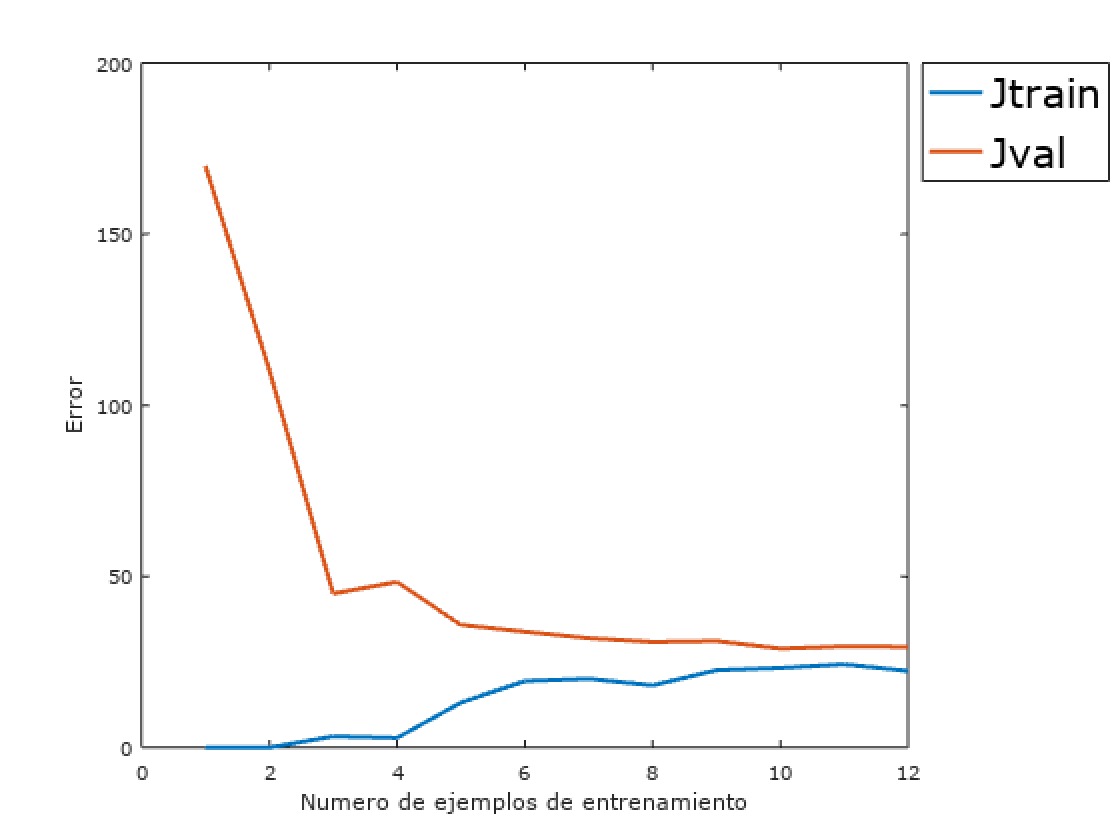


Y esta es la gráfica resultante:  
****

**CURVAS DE APRENDIZAJE**

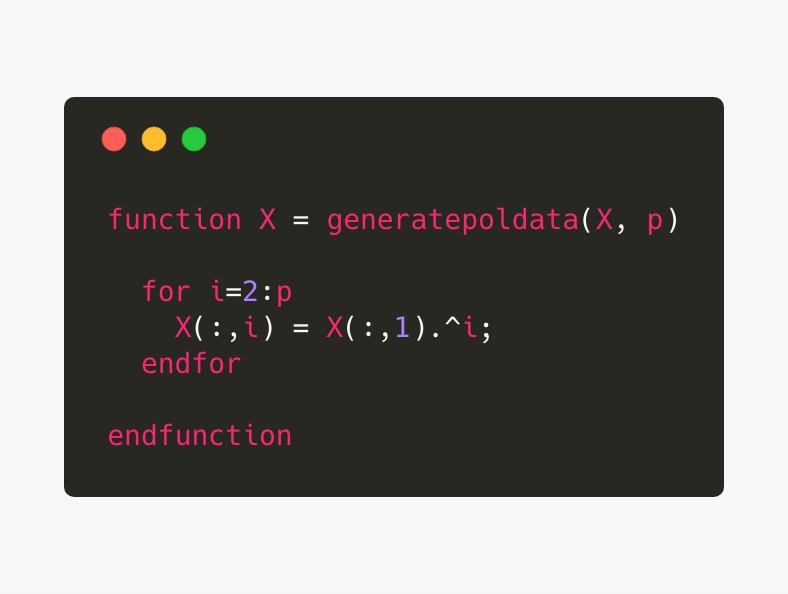
A continuación, hemos aplicado curvas de aprendizaje para ver cómo varía el error a medida que vas metiendo más datos de entrenamiento, tanto para los datos de entrenamiento como en los datos del cross validation. Para ello, hemos creado la función generarlc.m, que se encarga de calcular los thetas con el coste más bajo y de calcular este coste, añadiendo cada vez más datos de entrenamiento, dibujando una gráfica donde se ve cómo varian estos costes. Este es el código:

Esta es la gráfica obtenida:



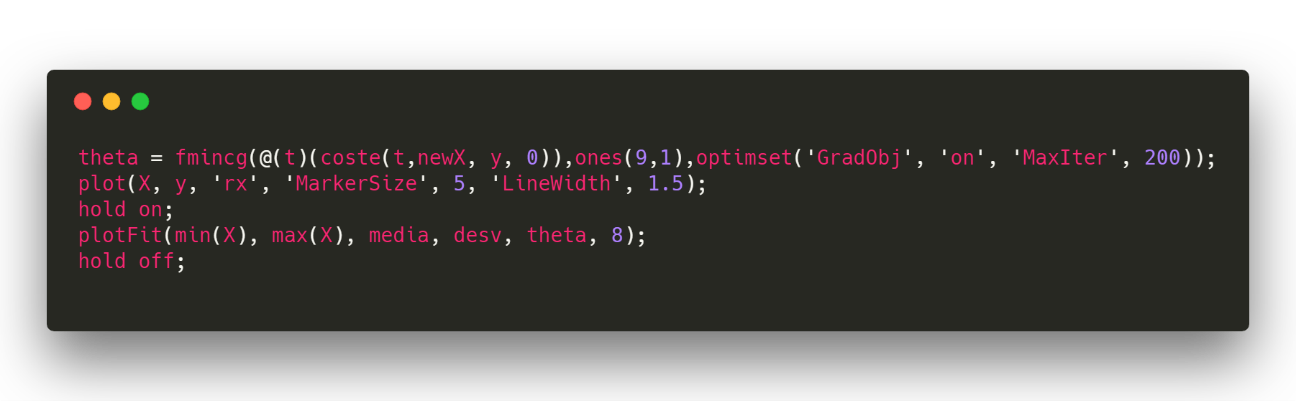
**REGRESIÓN POLINOMIAL**

En este apartado, lo primero que hemos tenido que hacer es una función que, dado una matriz de datos X y un número p, devuelva una matriz del mismo tamaño de X y con p columnas, cada una de ellas elevada a su índice de columna. Esta es la función que hemos creado y utilizado:



A continuación, creamos una matriz de 8 columnas con los datos de entrenamiento llamando a la función anterior, y además, la normalizamos usando la función proporcionada por la práctica llamada *featureNormalize.m.* Este es el código usado:

[newX, media, desv] = featureNormalize(generatepoldata(X,8));

Tras ejecutar la función *fmincg.m* con la nueva matriz X, nos proporciona las thetas con el error más bajo, y hemos representado la función generada, donde se aprecia claramente el overfiting existente. Para realizar esta función se ha usado la función *plotFit.m* proporcionada. Este es el código usado: 

Y esta es la gráfica generada:

