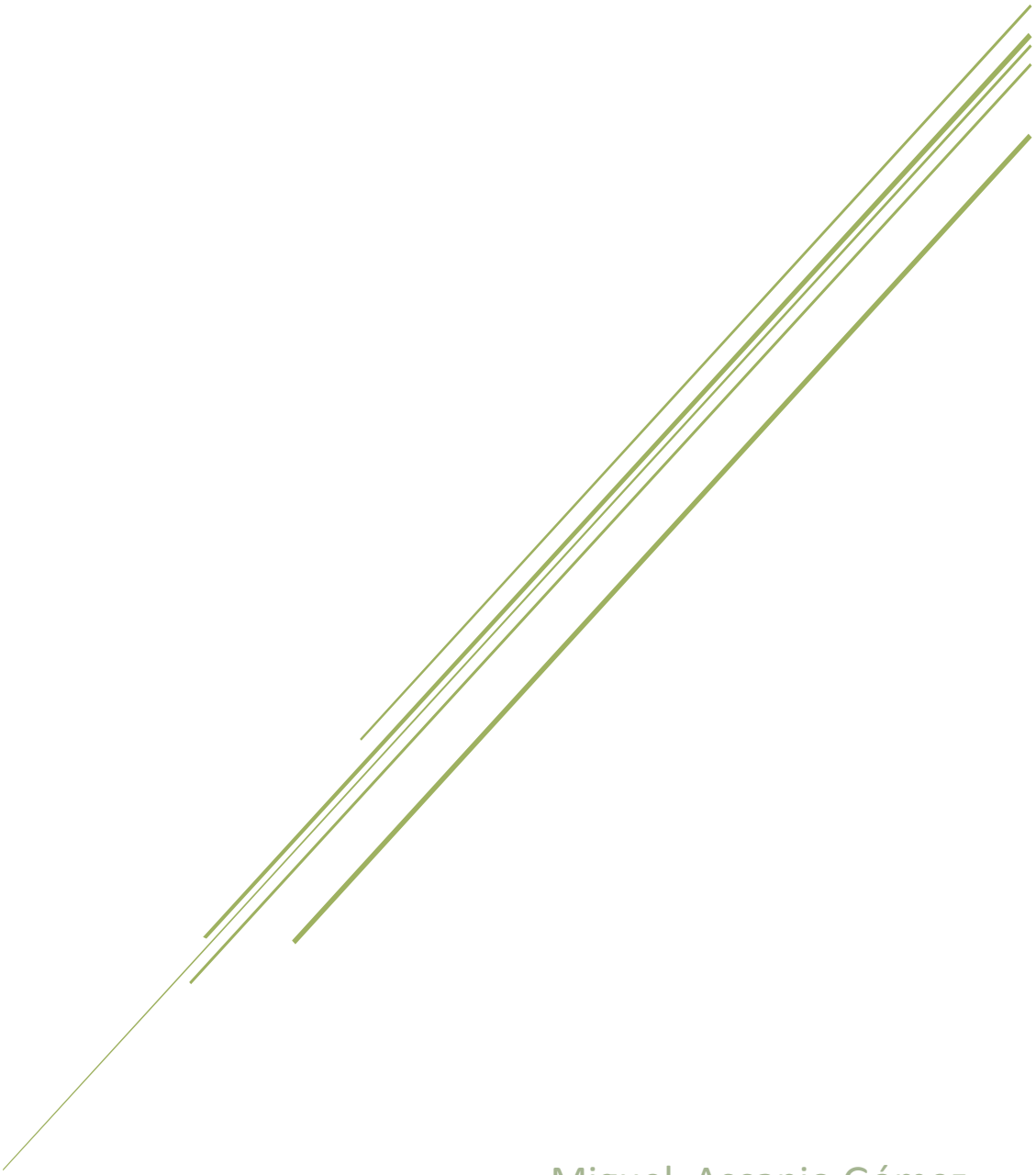
APRENDIZAJE AUTOMÁTICO

Práctica 5



Miguel Ascanio Gómez

Esther Ávila Benito

REGRESIÓN LINEAL REGULARIZADA

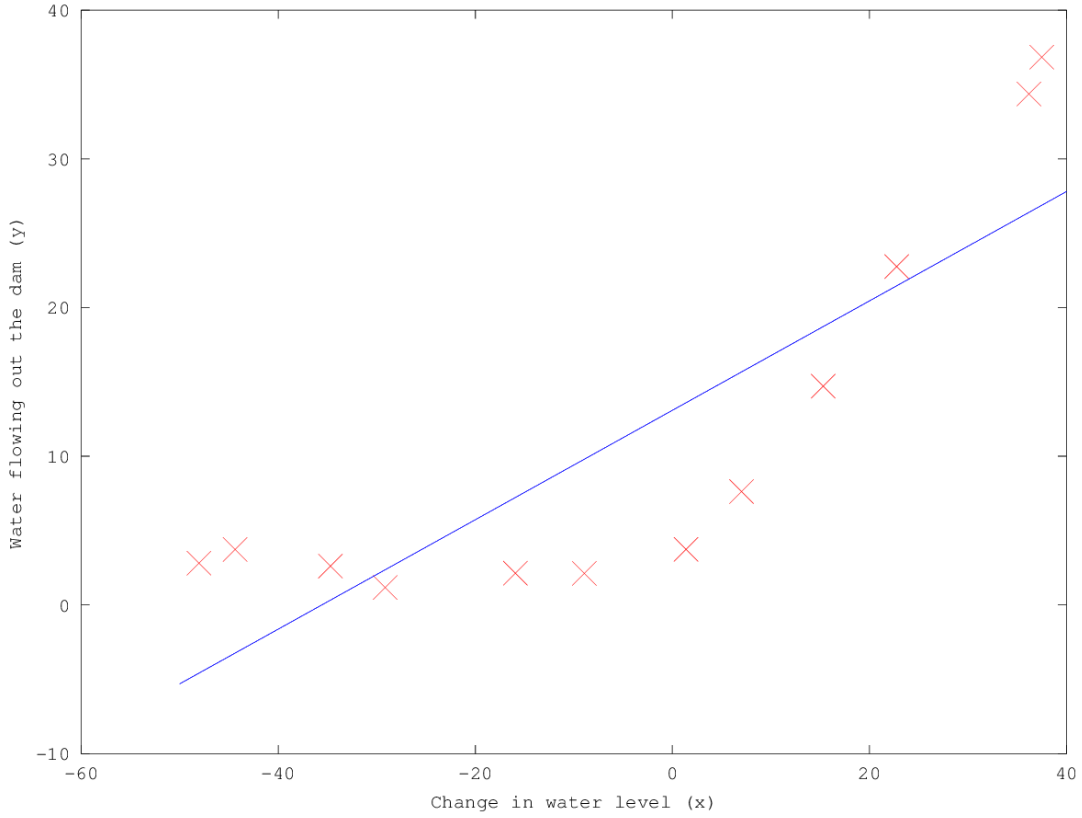
La función *coste* devuelve el coste y el gradiente de la regresión lineal regularizada para los ejemplos *X*.

```
function [J, grad] = coste(theta, X, y, lambda)
      warning("off", "Octave:broadcast");

      m = length(X(:,1));

      X = [ones(m, 1), X];
      valoresH = (X * theta);

      a = (valoresH - y) .^ 2;

      J = (1/2/m) * sum(a) + (lambda/2/m) * sum(theta.^2);

      # Sumandos
      a = (valoresH .- y) .* X;
      # Poner a cero para no incluir el primero
      theta(1) = 0;

      grad = ((1/m * sum(a)) .+ ((lambda/m) .* theta)')';

      warning("on", "Octave:broadcast");

endfunction
```

A partir del siguiente código, comprobamos que para un valor de *lambda = 1* y *theta = [1;1]* obtenemos un coste de 303.993.

```
load ex5data1.mat;

lambda = 0;
iters = 50;

thetaIni = [1;1];

funCoste = @(p)coste(p,X,y,lambda);

opciones = optimset ('Gradobj', 'on', 'MaxIter', iters);
[thetaProc, cost] = fmincg (funCoste, thetaIni, opciones);
cla;
hold on;
plot(X,y, 'x', 'markersize', 12, 'color', 'red');
plot(xRecta , thetaProc(1) .+ thetaProc(2) .* xRecta);

xlabel ("Change in water level (x)");
ylabel ("Water flowing out the dam (y)");

hold off;
```

Con la función *fmincg* proporcionada, se obtiene un vector *theta* que define una recta ajustada a los ejemplos de entrenamiento. El ajuste a algunos datos no es óptimo, eso se resolverá definiendo otro tipo de función (polinómica).



CURVAS DE APRENDIZAJE

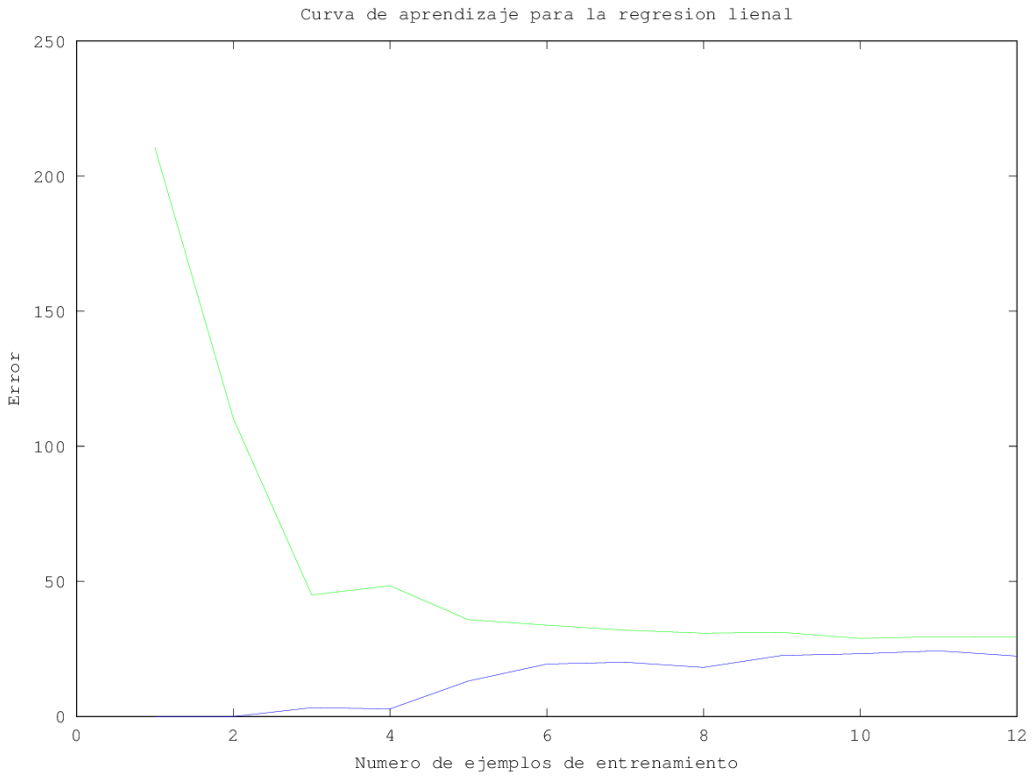
A continuación generaremos curvas de aprendizaje que se ajusten mejor a todos los ejemplos de entrenamiento utilizando subconjuntos de los mismos.

```
load ex5data1.mat;
lambda = 0;
maxIters = 50;
thetaIni = [1;1];

mEntren = length(X(:,1));
mPrueba = length(Xval(:,1));

funCoste = @(p)coste(p,X,y,lambda);
opciones = optimset ('Gradobj', 'on', 'MaxIter', maxIters);

for m = 1:mEntren
	funCoste = @(p)coste(p,X(1:m,:),y(1:m),lambda);
	[thetaProc, cost] = fmincg (funCoste, thetaIni, opciones);

	errorPrueba(m) = err(thetaProc, Xval, yval);
	errorEntren(m) = err(thetaProc, X(1:m,:),y(1:m));
endfor

cla;
hold on;
plot(1:mEntren, errorEntren, 'color', 'blue');
plot(1:mEntren, errorPrueba, 'color', 'green');

title ("Curva de aprendizaje para la regresion lienal");
ylabel ("Error");
xlabel ("Numero de ejemplos de entrenamiento");

hold off;
```

Función para calcular el coste (error):

```
function r = err(theta, X, y)
	warning("off", "Octave:broadcast");

	m = length(X(:,1));

	X = [ones(m, 1), X];
	valoresH = (X * theta);

	a = (valoresH - y) .^ 2;

	r = (1/2/m) * sum(a)

	warning("on", "Octave:broadcast");
endfunction
```

La gráfica generada es la siguiente:



Se observa que a mayor número de ejemplos de entrenamiento, el error con los ejemplos de validación cae rápidamente, mientras que el error con los propios elementos de entrenamiento sube.

REGRESIÓN POLINOMIAL

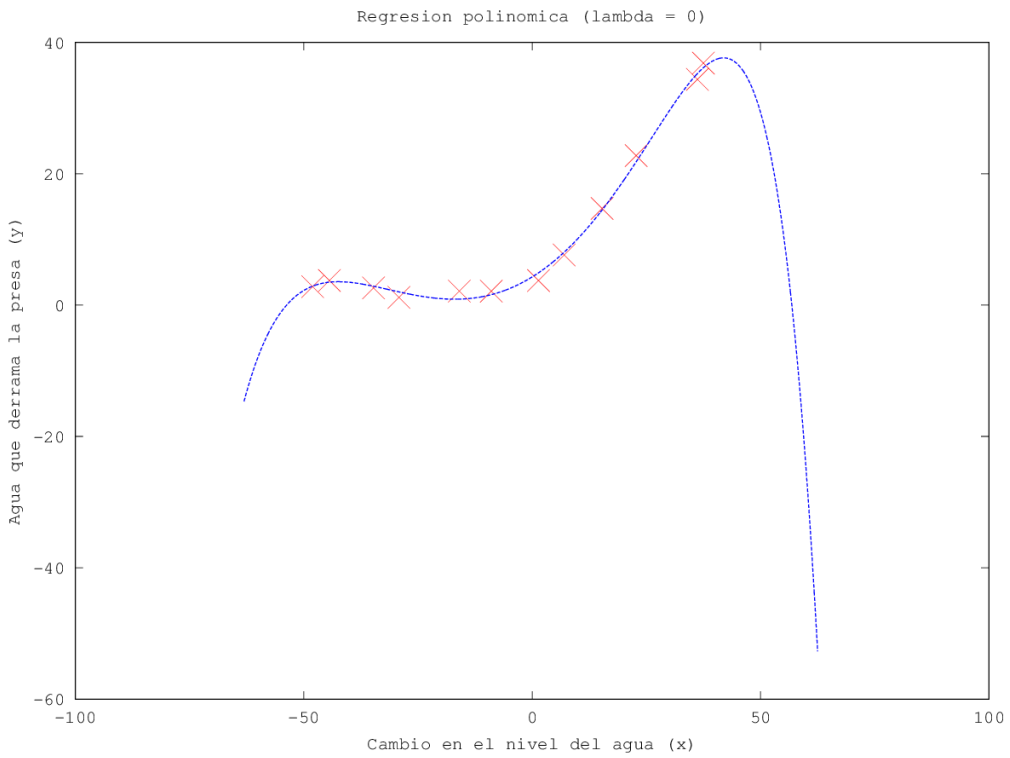
El objetivo de este apartado es conseguir un mayor ajuste a los ejemplos de entrenamiento. Para ello, usaremos unos nuevos ejemplos generados a partir de la siguiente función:

```
function r = genera(X, p)

r = X;
for i=2:p
  r = [r X.^i];
end

endfunction
```

Posteriormente, hay que normalizar los atributos para reducir el rango entre todos ellos. Esta normalización ha sido realizada con la función *featureNormalize* proporcionada con la práctica.

A continuación, se vuelve aplicar regresión lineal a los nuevos datos vara obtener el valor *theta* que minimiza el error con un valor *lambda = 0 (sin regularización)*.

```
clear;
load ex5data1.mat;

lambda = 0;
maxIters = 50;
p = 8;
thetaIni = ones(p+1,1);

mEntren = length(X(:,1));
mPrueba = length(Xval(:,1));

X_gen = genera(X,p);
[X_norm, mu, sigma] = featureNormalize(X_gen);


funCoste = @(t)coste(t,X_norm,y,lambda);
opciones = optimset ('Gradobj', 'on', 'MaxIter', maxIters);

[thetaProc, cost] = fmincg (funCoste, thetaIni, opciones);
cla;
hold on;
plot(X,y, 'x', 'markersize', 12, 'color', 'red');
plotFit(min(X),max(X), mu, sigma, thetaProc, p);

xlabel ("Cambio en el nivel del agua (x)");
ylabel ("Agua que derrama la presa (y)");
title ("Regresion polinomica (lambda = 0)");
```

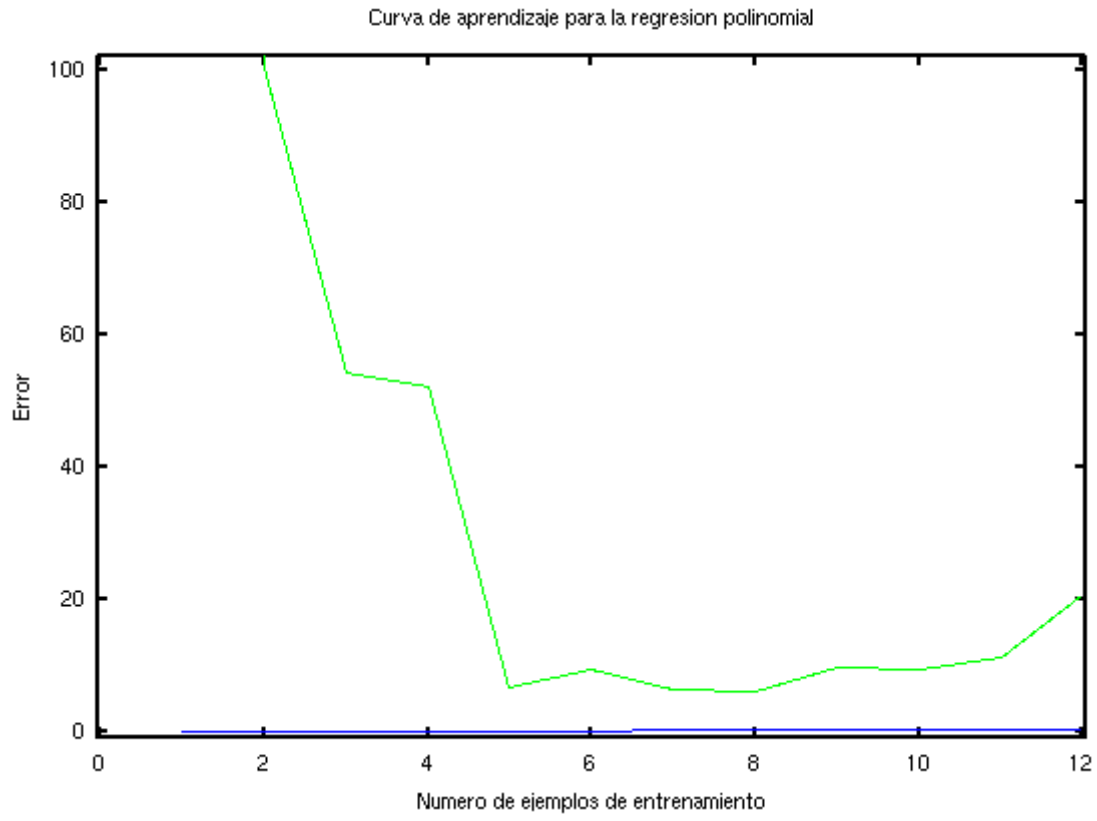


Al aumentar el número de iteraciones de fmincg, se observa que la gráfica cambiaba notablemente: pasaba igual por los ejemplos de entrenamiento (las X), sin embargo por la derecha crecía, no decrecía.

A continuación se vuelven a generar las curvas de aprendizaje para la hipótesis polinomial. Para ello se vuelve a aplicar regresión lineal a subconjuntos de los ejemplos de validación. Éstos previamente han sido transformados y normalizados al igual que los ejemplos de entrenamiento.

```
clear;
load ex5data1.mat;

lambda = 0;
maxIters = 100;
p = 8;
thetaIni = ones(p+1,1);

mEntren = length(X(:,1));
mPrueba = length(Xval(:,1));

X_gen = genera(X,p);
[X_norm, mu, sigma] = featureNormalize(X_gen);

Xval_gen = genera(Xval, p);

Xval_norm = Xval_gen;

Xval_norm = bsxfun(@minus, Xval_norm, mu);
Xval_norm = bsxfun(@rdivide, Xval_norm, sigma);

funCoste = @(p)coste(p,X,y,lambda);
opciones = optimset ('Gradobj', 'on', 'MaxIter', maxIters);


for m = 1:mEntren
```

```
        funCoste = @(p)coste(p,X_norm(1:m,:),y(1:m),lambda);
        [thetaProc, cost] = fmincg (funCoste, thetaIni, opciones);

        errorPrueba(m) = err(thetaProc, Xval_norm, yval);
        errorEntren(m) = err(thetaProc, X_norm(1:m,:),y(1:m));

endfor

cla;
hold on;
plot(1:mEntren, errorEntren, 'color', 'blue');
plot(1:mEntren, errorPrueba, 'color', 'green');

title ("Curva de aprendizaje para la regresion polinomial");
ylabel ("Error");
xlabel ("Numero de ejemplos de entrenamiento");

hold off;
```

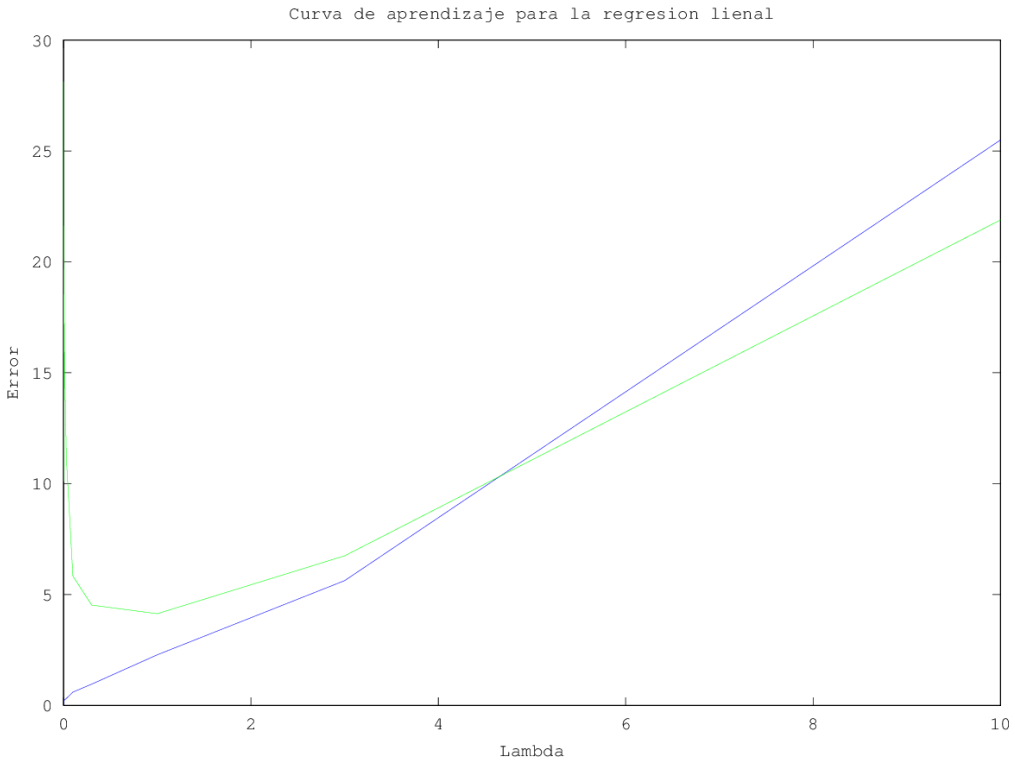


SELECCIÓN DEL PARÁMETRO *LAMBDA*

El objetivo de este apartado es encontrar el valor de *lambda* que minimice el error sobre los ejemplos de validación. Para ello se muestra los valores de error para los ejemplos de entrenamiento y un conjunto de ejemplos de validación. Se harán pruebas para los siguiente valores de lambda: 0, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3, 10.

```
clear;
load ex5data1.mat;

lambda = 0;
maxIters = 100;
p = 8;
thetaIni = ones(p+1,1);

mEntren = length(X(:,1));

X_gen = genera(X,p);
[X_norm, mu, sigma] = featureNormalize(X_gen);

Xval_gen = genera(Xval, p);

Xval_norm = Xval_gen;

Xval_norm = bsxfun(@minus, Xval_norm, mu);
Xval_norm = bsxfun(@rdivide, Xval_norm, sigma);

opciones = optimset ('Gradobj', 'on', 'MaxIter', maxIters);
lambda = [0, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3, 10];
for i = 1:length(lambda)

      funCoste = @(t)coste(t,X_norm,y,lambda(i));
      [thetaProc, cost] = fmincg (funCoste, thetaIni, opciones);

      errorPrueba(i) = err(thetaProc, Xval_norm, yval);
      errorEntren(i) = err(thetaProc, X_norm,y);

endfor

cla;
hold on;
size(lambda)
size(errorEntren)
plot(lambda, errorEntren, 'color', 'blue');
plot(lambda, errorPrueba, 'color', 'green');

title ("Curva de aprendizaje para la regresion lineal");
ylabel ("Error");
xlabel ("Lambda");

hold off;
```

Se obtiene la siguiente curva de aprendizaje:



Se observa que el lambda óptimo es 1. Probando el coste de la función respecto a los datos de testeo XTest e Ytest, habiendo sido modificados para ser de grado 8 y normalizado con la mu y sigma de X, con el siguiente código:

```
lambda = 1;
maxIters = 100;
p = 8;
thetaIni = ones(p+1,1);
mEntren = length(X(:,1));

X_gen = genera(X,p);
[X_norm, mu, sigma] = featureNormalize(X_gen);

Xtest_gen = genera(Xtest, p);
Xtest_norm = Xtest_gen;
Xtest_norm = bsxfun(@minus, Xtest_norm, mu);
Xtest_norm = bsxfun(@rdivide,Xtest_norm, sigma);

opciones = optimset ('Gradobj', 'on', 'MaxIter', maxIters);
funCoste = @(t)coste(t,X_norm,y,lambda);
[thetaProc, cost] = fmincg (funCoste, thetaIni, opciones);

errorPrueba = err(thetaProc, Xtest_norm, ytest)
```

Se obtiene un error de 2.8657
Nota: se obtienen unos resultados ligeramente diferentes a los del enunciado tanto en este apartado como en el anterior