Práctica 5: Regresión lineal regularizada: sesgo y varianza

Grupo 13:
David Ortiz Fernández.
Andrés Ortiz Loaiza.

Esta práctica está orientada a estudiar el sesgo y la varianza en la regresión lineal regulariza, y ver los efectos de overfiting y el underfiting que tiene al elegir distintos valores del parámetro lamba en la regularización y del grado del polinomio. También visualizaremos las curvas de aprendizaje. Para ello hemos seguido los siguientes pasos.

Como siempre comenzaremos cargando los datos. Siguiendo el guion de la práctica hemos desarrollado una función que se encarga de calcular el coste y el gradiente de la regresión lineal regularizada (CosteGradRegu).

Tras mostrar los datos por pantalla hemos realizado las comprobaciones que nos pide el guion siendo estos: Coste con theta = [1 ; 1]: 303.993192 y Gradiente con theta = [1 ; 1]: [-15.303016; 598.250744].

Usando la función fmincg proporcionada por la práctica y con 200 iteraciones, hemos entrenado nuestro modelo y hemos obtenido un vector θ que de una recta ajustada a los datos de X e y similar a la que aparece en el guion, la cual esta adjunta más adelante en esta memoria, y como observamos en ella no es un modelo que ajuste muy bien la prediccion.

En siguiente lugar, hemos generado la curva de aprendizaje para ver cómo varía el error a medida que aumentan datos de entrenamiento, tanto para los datos de entrenamiento como en los datos del cross validation. Para generar las curvas de aprendizaje, se ha entrenado nuevamente por regresión lineal, utilizando los diferentes grupos de datos que hemos mencionado, pero con el termino de regularizacion igual a 0 (lambda=0).

En el apartado de la regresión polinomial, hemos creado 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 (función datosPoli). Tras esto hemos entrenado nuevamente el modelo y hemos generado la gráfica correspondiente, donde se aprecia el correcto ajuste del modelo a los datos.

Finalmente comprobamos que se ha ajustado correctamente el modelo aplicando los datos de test a nuestra hipótesis, en base a distintas lambdas. Posteriormente se ha visualizado el error de entrenamiento y validación de acorde con diferentes valores del termino de regularización.

El error de validación para lambda=3 es 3.822907, cumpliendo las indicaciones del guion de la practica.

Flujo principal de la práctica

```
clear ;
close all;
load ('ex5data1.mat');
m = size(X, 1);
%utilizamos who para ver que variables se han incorporado al entorno
printf('variables se han incorporado al entorno.\n');
%llamada a la función who
who

% Plot
plot(X, y, 'rx', 'MarkerSize', 10, 'LineWidth', 2);
xlabel('Change in water level (x)');
ylabel('Water flowing out of the dam (y)');

printf('Presione Entrer para continuar.\n');
pause;

lambda = 1;
theta = [1 ; 1];

%Funcion de coste y gradiente regularizado
[J, grad] = CosteGradRegu([ones(m, 1) X], y, theta, lambda);

printf('Coste con theta = [1 ; 1]: %f ', J);
printf('Gradiente con theta = [1 ; 1]:  [%f; %f] ',grad(1), grad(2));

printf('\n Presione Entrer para continuar.\n');
pause;

% Entrenamiento regresion lineal con lambda = 0
lambda = 0;
[theta] = entrenamientoLinReg([ones(m, 1) X], y, lambda);

%  Plot
plot(X, y, 'rx', 'MarkerSize', 10, 'LineWidth', 1.5);
xlabel('Change in water level (x)');
ylabel('Water flowing out of the dam (y)');
hold on;
plot(X, [ones(m, 1) X]*theta, '-', 'LineWidth', 2)
hold off;

printf('\n Presione Entrer para continuar.\n');
pause;

%2. Curvas de aprendizaje
```

```
lambda = 0;
%lambda = 1;
%lambda = 100;
[error,errorVali]=curvaAprendizaje([ones(m,1) X],y,[ones(size(Xval,1),1) Xval],yval,lambda);

plot(1:m, error, 1:m, errorVali);
title('Curva de aprendizaje para la regresión lineal')
legend('Entrenamiento', 'Validacion')
xlabel('Numero de ejemplos de entrenamiento')
ylabel('Error')

printf('\n Presione Entrer para continuar.\n');
pause;

%3. Regresión polinomial
p = 8;
% Normalizar
Xpoli = datosPoli(X, p);
[Xpoli, mu, sigma] = featureNormalize(Xpoli);  % Normalize

% Añadir unos
Xpoli = [ones(m, 1), Xpoli];

%entrenamiento
lambda = 0;
[theta] = entrenamientoLinReg(Xpoli, y, lambda);
figure(1);
%pot
plot(X, y, 'rx', 'MarkerSize', 10, 'LineWidth', 2);
plotFit(min(X), max(X), mu, sigma, theta, p);
xlabel('Cambio en el nivel del agua (x)');
ylabel('Agua que derrama la presa (y)');
title (sprintf('Regresión Polinómica (lambda = 0)'));

printf('\n Presione Entrer para continuar.\n');
pause;
%para calcular el error sobre los ejemplos de validación Xval debes aplicarles
%la misma transformación que a los de entrenamiento,generando las pot. desde 1
%hasta p y normalizándolas luego usando las medias y desviaciones estándar
%calculadas para los ejemplos de entrenamiento
% codigo inspirado en plotfit
X_poly_val = datosPoli(Xval, p);
X_poly_val = bsxfun(@minus, X_poly_val, mu);
X_poly_val = bsxfun(@rdivide, X_poly_val, sigma);
 % Añadimos unos
```

```
X_poly_val = [ones(size(X_poly_val, 1), 1), X_poly_val];

%A continuación, puedes probar el efecto que tiene el término de regularización
%repitiendo el proceso para ? = 1 y ? = 100.
% plots
lambda = 0;
%lambda = 1;
%lambda = 100;
[error, errorVali] = curvaAprendizaje(Xpoli, y, X_poly_val, yval, lambda);
figure(2);
plot(1:m, error, 1:m, errorVali);
title(sprintf('Curva de aprendizaje para la regresión polinomial(lambda = 0)'));
xlabel('Numero de ejemplos de entrenamiento')
ylabel('Error')
legend('Entrenamiento', 'Validación')

printf('\n Presione Entrer para continuar.\n');
pause;

%4. Selección del parámetro ?
lambdaV = [0 0.001 0.003 0.01 0.03 0.1 0.3 1 3 10]';
[error, errorVali] =validacion(Xpoli, y, X_poly_val, yval,lambdaV);

figure(3);
plot(lambdaV, error, lambdaV, errorVali);
legend('Entrenamiento', 'Validación');
xlabel('lambda');
ylabel('Error');
printf('\n Presione Entrer para continuar.\n');
pause;

printf('Error de validacion para lambda=3: %f\n', errorVali(9));
```

Funciones:

CosteGradRegu

```
%a función que devuelvael coste y el gradiente de la regresión lineal
%regularizada a partir de los valores de entrada X, salida y, el vector de
%parámetros ? y el valor de ?
function [J, grad] = CosteGradRegu(X, y, theta, lambda)

J = 0;
grad = zeros(size(theta));
m = length(y);
%calcul ode hipotesis
h = X*theta;
dif = h - y;

thetaReg = [0 ; theta(2:end, :)];

%calculo de función de coste
p = lambda*(thetaReg'*thetaReg);

J = (dif'*dif)/(2*m) + p/(2*m);

% calculo del gradiente
grad = (X'*dif+lambda*thetaReg)/m;

grad = grad(:);

endfunction
```

entrenamientoLinReg

Función de entrenamiento

```
function [theta] = entrenamientoLinReg(X, y, lambda)

initial_theta = zeros(size(X, 2), 1);

costFunction = @(t) CosteGradRegu(X, y, t, lambda);

options = optimset('MaxIter', 200, 'GradObj', 'on');

% minimización usando fmincg
theta = fmincg(costFunction, initial_theta, options);

endfunction
```

curvaAprendizaje

Mediante esta función generamos los valores de error de entrenamiento y de validación para posteriormente usarlos al pintar la gráfica de aprendizaje polinomial y lineal.

```
function [error, errorVali] = curvaAprendizaje(X, y, Xval, yval, lambda)
m = size(X, 1);
% error del resultado aplicado sobre esemismo subconjunto, y el error al
% clasificar a los ejemplos del conjunto de validación
error = zeros(m, 1);
errorVali   = zeros(m, 1);

for i= 1:m
    theta = entrenamientoLinReg(X(1:i,:), y(1:i), lambda);
    %introducimos el valor lambda=0
    error(i) = CosteGradRegu(X(1:i,:), y(1:i), theta, 0);
    errorVali(i)   = CosteGradRegu(Xval, yval, theta, 0);
endfor

endfunction
```

datosPoli

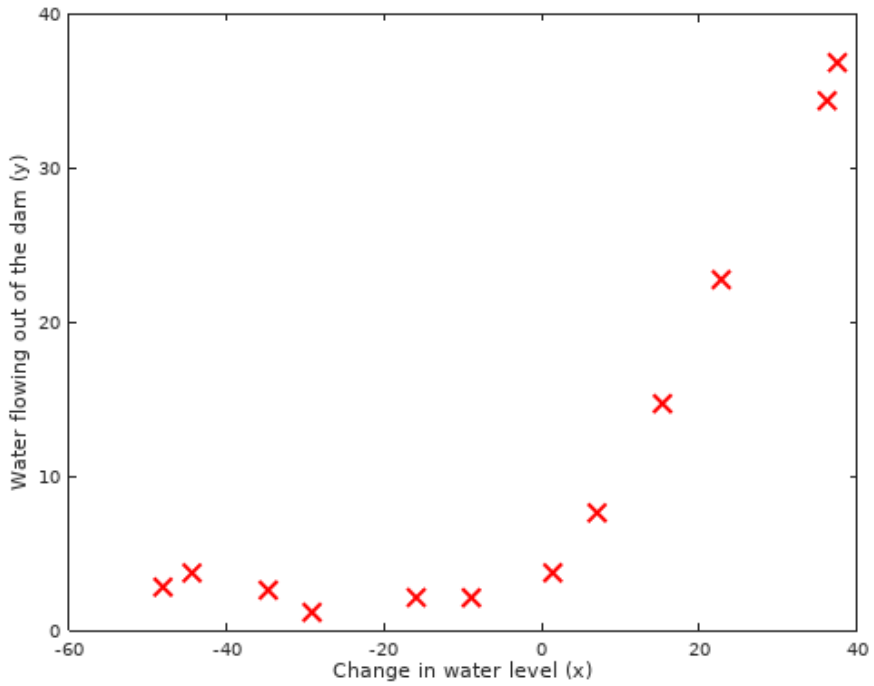
```
%Esta función recibirá una matriz X de dimensión m × 1 y un número p, y
%devolverá otra matriz de dimensión m × p que en la primera columna contenga
%los valores de X, en la segunda el resultado de calcular X.^2, en la tercera
%X.^3, y así sucesivamente.
function Xpoli = datosPoli(X, p)

Xpoli = zeros(numel(X), p);
Xpoli(:,1) = X;

for i=2:p
    Xpoli(:,i) = X.*Xpoli(:,i-1);
endfor


endfunction
```

validación

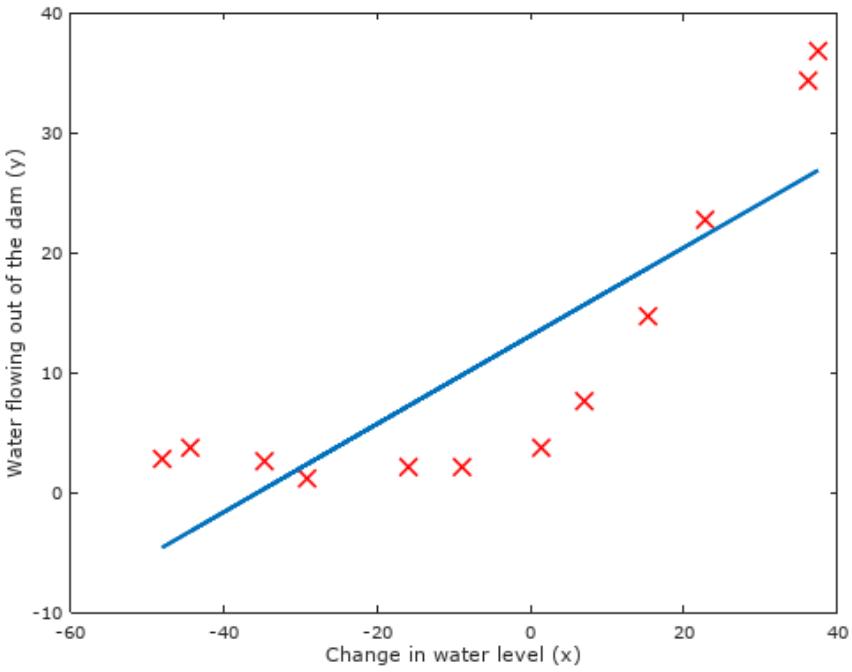
```
%funcion que genera los valores de error de entrenamiento y validacion para
%un conjunto lambda que entra como parametro
function [error, errorVali] = validacion(X, y, Xval, yval,lambdaV)

% inicialización de variables a retornar
error= zeros(length(lambdaV), 1);
errorVali = zeros(length(lambdaV), 1);

%generacion de los valores de error para todos los lambda dados
for i = 1:length(lambdaV)
    lambda = lambdaV(i);
    theta = entrenamientoLinReg(X,y,lambda);
    error(i) = CosteGradRegu(X,y,theta,0);
    errorVali(i)    = CosteGradRegu(Xval,yval,theta,0);
endfor

endfunction
```

Resultados:

Representación de los datos

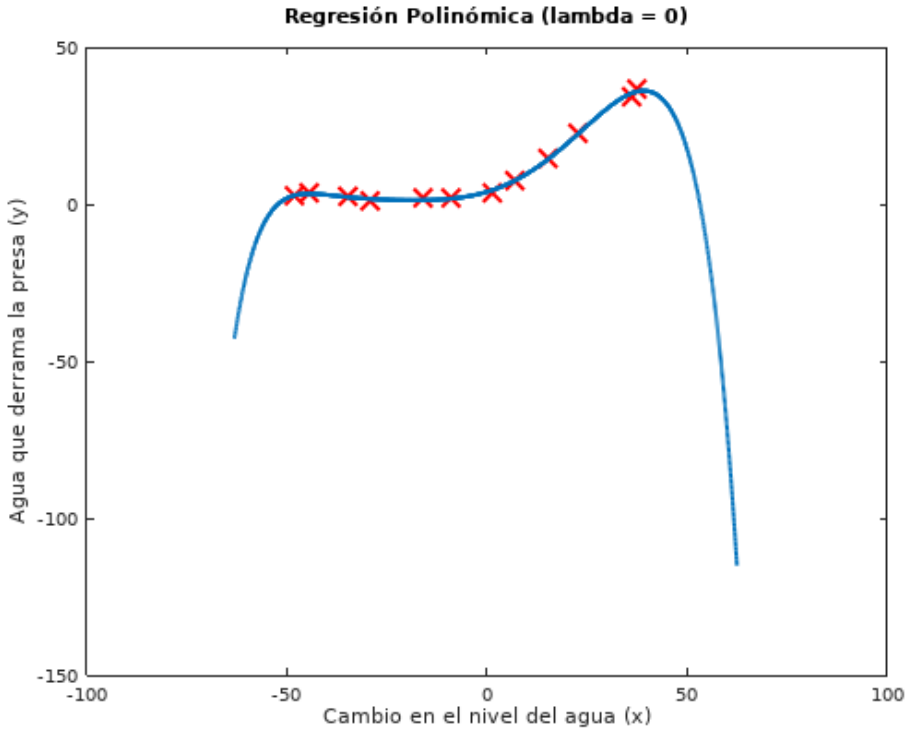


Primer ajuste en el que se aprecia que la recta no se ajusta bien a los datos

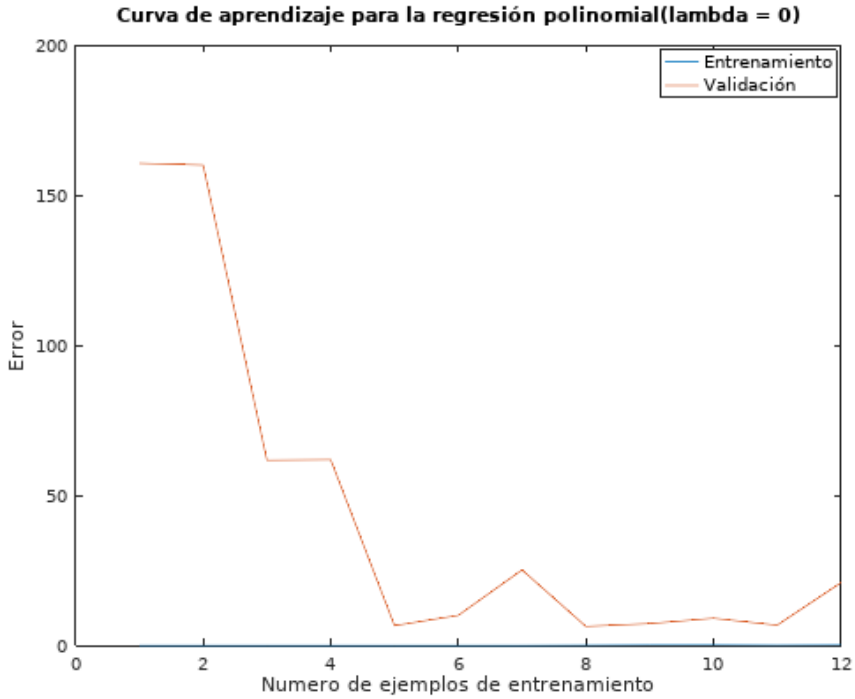


Coste con theta = [1 ; 1]: 303.993192
Gradiente con theta = [1 ; 1]: [-15.303016; 598.250744]

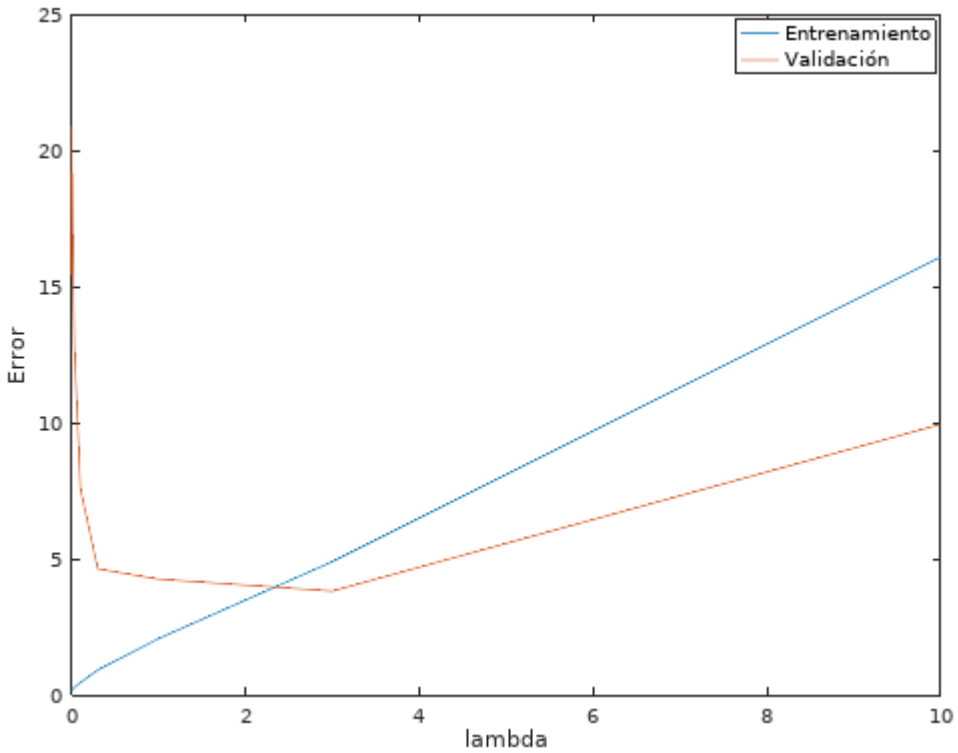
Curva que representa a la hipótesis aprendida



Gráfica donde se muestra que la hipótesis está sobre-ajustada a los ejemplos de entrenamiento:



Gráfica con los valores de error para los ejemplos de entrenamiento y los de validación tras aplicar el método de regresión polinomial.



Error de validación para lambda=3: 3.822907