# Práctica 5: Regresión lineal regularizada: sesgo y varianza

Esta práctica consiste en aplicar la regresión lineal regularizada. Esto es que una hipótesis sesgada no podremos clasificar bien los ejemplos de entrenamiento. Por ello luego ajustaremos los datos de entrenamiento a un grado superior, para que los ejemplos se puedan clasificar mejor.

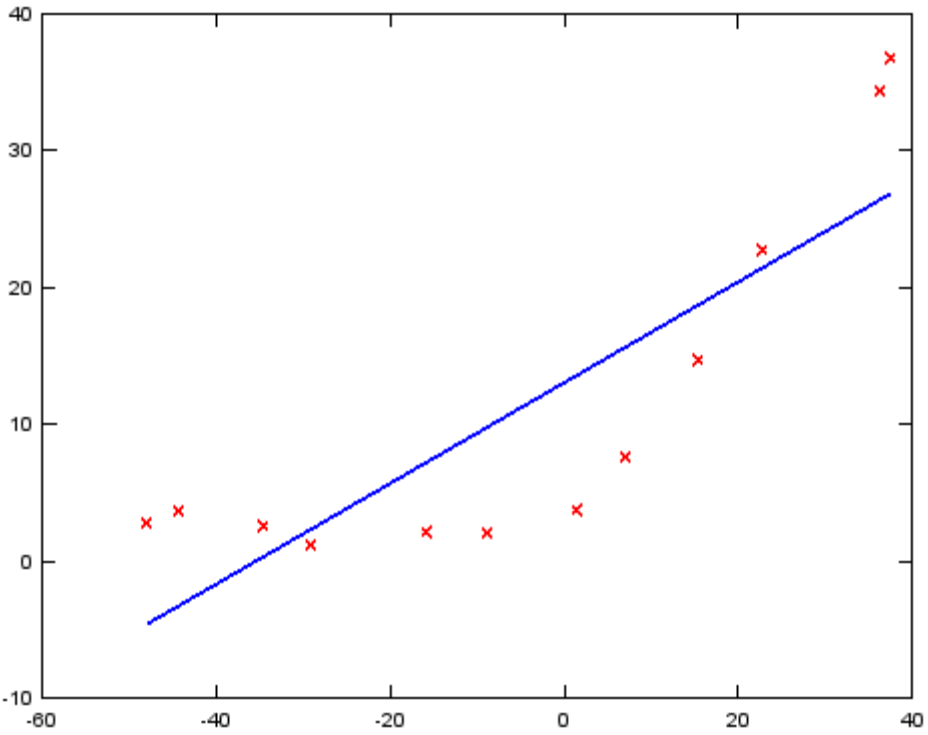
Después haremos pruebas con el término de regularización (alpha) para ver cuál es el mejor valor, ya que un sobreajuste no sería bueno para clasificar ejemplos de entrenamiento posteriores. Así que evaluaremos la hipótesis generada sobre los ejemplos de entrenamiento con un segundo conjunto que no utilizaremos para generar la hipótesis, pudiendo comprobar cómo se ajusta a estos nuevos datos.

## Regresión lineal regularizada

Lo primero que debemos hacer es visualizar los datos que vamos a emplear. Como sólo tenemos 2 variables, es fácil realizar la visualización. Primero tendremos que cargar los datos de entrenamiento desde 'ex5data1.mat', y después los pintaremos mediante la función plot.

Después de ello realizaremos la regresión lineal regularizada, para la que implementaremos la función de “costeRegresionLinealRegularizada” mediante las fórmulas indicadas. Esta función nos devolverá el coste y gradiente de la regresión lineal a partir de los valores de entrada. En la implementación habrá que tener cuidado en la diferenciación del gradiente para j=0 y j>=1.

Después tendremos que utilizar esta función para buscar el valor de theta que minimice el coste, implementando la función "entrenaRegularizacionLineal". El resultado de esta función con un landa de 0 es el siguiente:



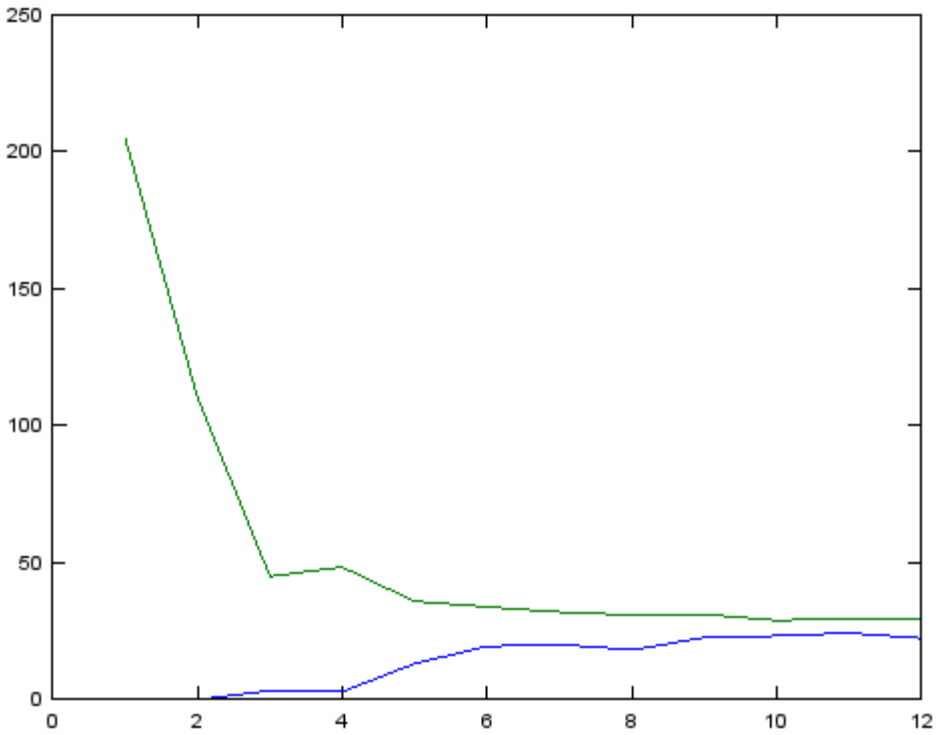

Como podemos ver en la imagen, aunque haya minimizado el coste, la recta no se ajusta a los ejemplos de entrenamiento, porque necesitaría de más parámetros para ello. Es algo que arreglaremos mediante la regresión polinomial.

## Curvas de aprendizaje

Como en un ejemplo en el que tengamos muchos atributos no podremos realizar una representación gráfica tan sencilla, en estos casos utilizamos las curvas de aprendizaje para ver las situaciones de sesgo (sub-ajuste) y varianza (sobre-ajuste).

Para ello, repetiremos el entrenamiento con distintos subconjuntos de los datos de entrenamiento, evaluando el error del resultado del

entrenamiento y el error de clasificar nuevos datos. Debería darse la situación de que cuanto menor es el error de uno, mayor será el error de la otra. De esta forma se generará una gráfica similar a esta, en la que se puede observar que las 2 gráficas “convergen” en cierto punto.



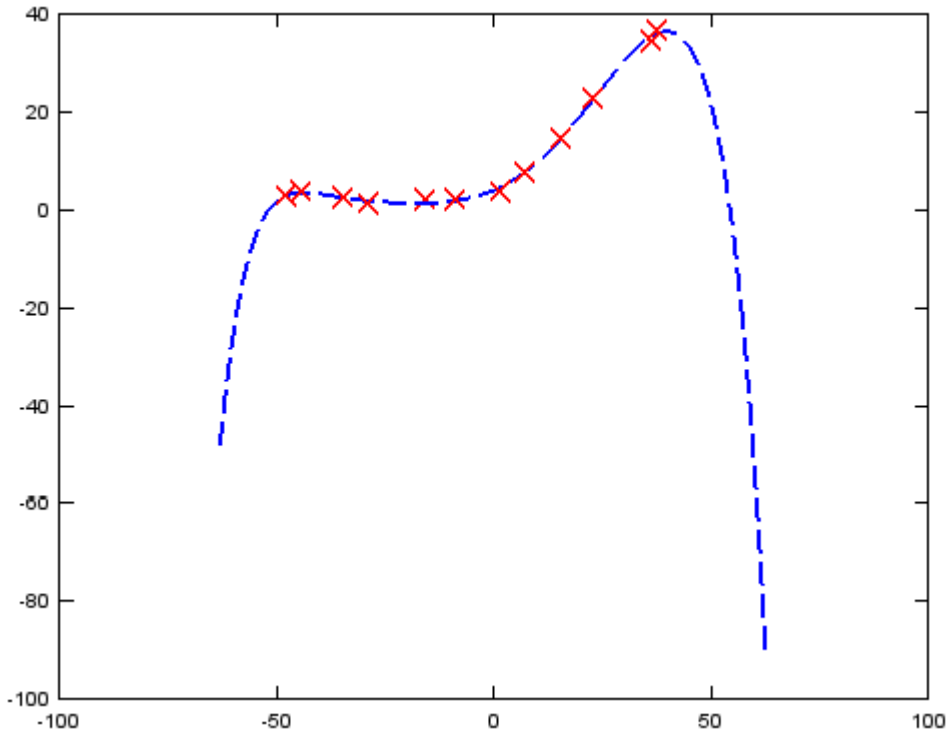

Este tipo de gráfica se debe precisamente a que el aprendizaje está sesgado por el número de parámetros empleados en la hipótesis.

## Regresión polinomial

Para que el ajuste sea mayor, emplearemos como hipótesis un polinomio en vez de una recta.

Para ello implementaremos una función “elevarPotenciaAtributos” que nos servirá para generar nuevos datos de entrenamiento a partir de los datos originales. Como generaremos potencias de 2 habrá mucha diferencia entre unos y otros, por lo que habrá que normalizar los atributos antes de poder utilizarlos.

Una vez hayamos generado los nuevos atributos realizaremos los mismos pasos que en el punto anterior, generando esta vez una hipótesis que será una curva:

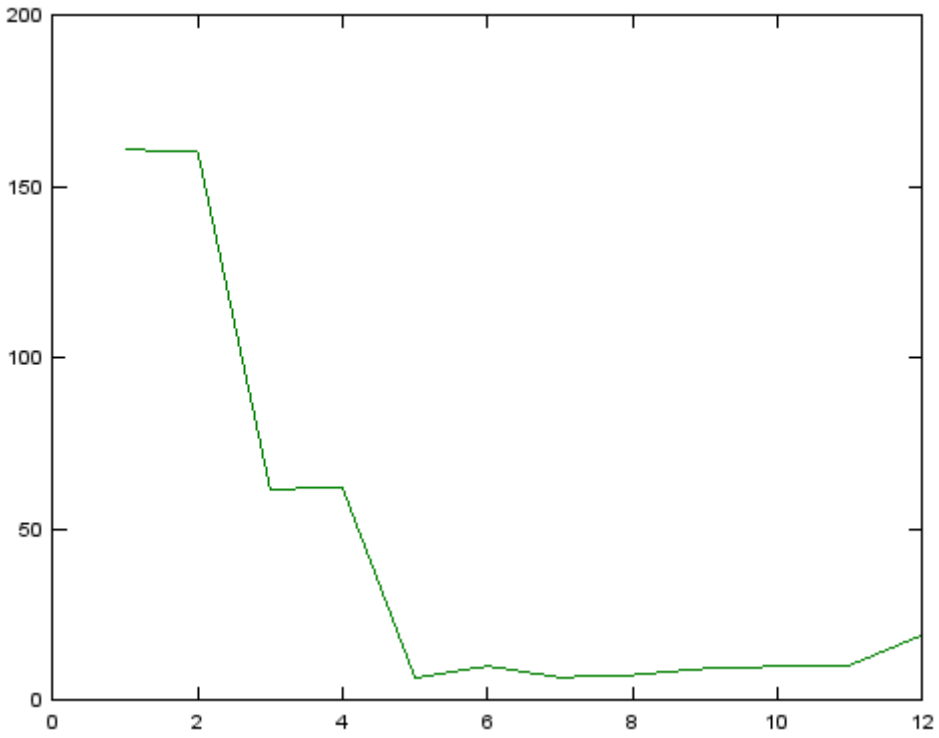



Ahora se puede ver que esta nueva hipótesis polinomial se ajusta bastante mejor a los ejemplos de entrenamiento.

## Curvas de aprendizaje

Como en el ejemplo anterior, debemos generar las curvas de aprendizaje para ver cómo se comportan ante esta nueva regresión polinomial.

Podemos ver en la gráfica que la hipótesis está sobre-ajustada (varianza) a los ejemplos de entrenamiento, ya que tiene un error de prácticamente cero. Esto puede no ser beneficioso para los ejemplos posteriores:

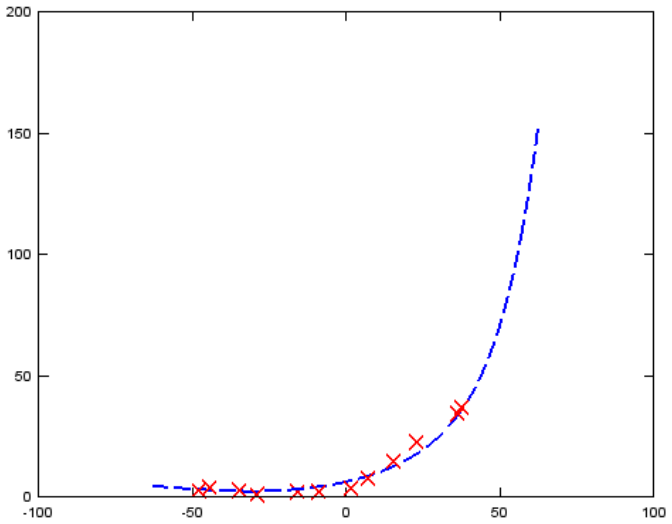


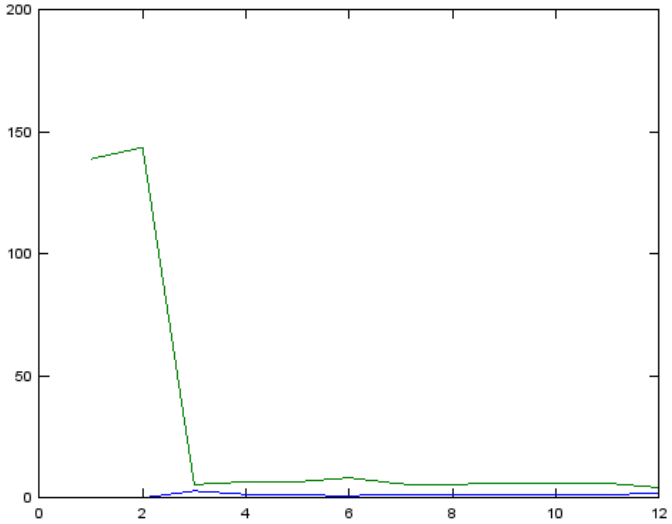

Podemos ver que el error es casi cero.

Una de las cosas importantes es la elección del parámetro landa, ya que si este no es el correcto se puede producir un sobreajuste o sesgo.

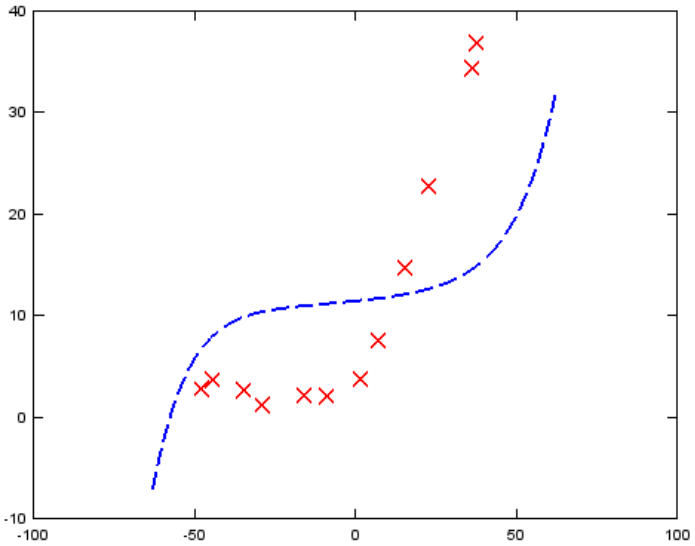
Podemos ver 2 casos como ejemplo:

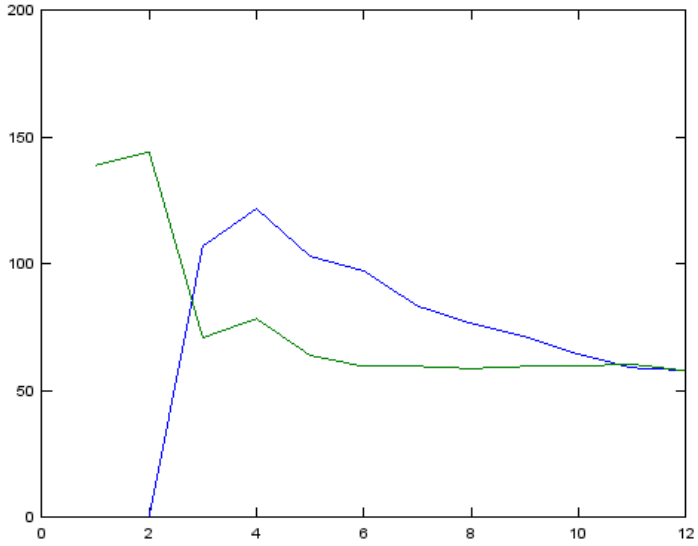
Si el landa=1, la curva se ajusta bastante bien, aunque quizá no tanto como con el landa 0. Algo a favor es que parece que aunque el error sobre los datos de entrenamiento no sea cero, el error sobre los datos de validación se reduce bastante más.




Si el landa=100, la curva se ajusta mucho menos a los datos de entrenamiento, pero a pesar de ello esto hace que ambos errores aumenten bastante más en comparación con lo original.



## Elección del parámetro landa

Como hemos podido comprobar, la elección del parámetro landa tiene mucha importancia en el resultado, por lo que es muy importante elegir el valor que minimice el error sobre los ejemplos de validación.

Para ello repetiremos el experimento para varios valores de landa, y nos quedaremos con el de menor error. En este caso el mejor valor de landa es de 3, que genera el siguiente error: