Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Aprendizaje automático

Practica 1: Regresión

Autor:
Nerea Gallego Sánchez (801950)

3 de Marzo de 2021

Índice

1. Introducción y objetivos

Esta práctica consiste en aprender y consolidar los conceptos sobre regresión vistos en clase. Por lo tanto, el objetivo de la misma es aplicar las técnicas de regresión lineal en casos reales.

El caso de estudio es la predicción del precio de unos pisos según su superficie (medida en m^2) y la cantidad de habitaciones que tiene. Para ello, se proporcionan un conjunto de datos de entrenamiento y un conjunto de datos de test.

Los distintos algoritmos que se van a utilizar son: regresión monovariable utilizando la ecuación normal, regresión multivariable utilizando la ecuación normal, regresión monovariable utilizando descenso de gradiente, regresión multivariables utilizando descenso de gradiente y regresión robusta utilizando el coste de Huber. Para cada algoritmo, se utiliza un conjunto de datos de entrenamineto con los que se entrena el modelo y a continuación se utiliza un conjunto de datos de test para evaluar el modelo. De la misma manera, se comparan los algoritmos entre sí mediante los resultados obtenidos.

2. Estudio previo

Supongo theta inicial

```
function T = descensoDeGradiente(X, theta, y, alpha)
    r = X*theta - y
    grad = X'*r
    theta2 = theta - alpha.*grad
    while (mean(abs(theta - theta2)) > 0.001)
        theta = theta2
        r = X*theta - y
        grad = X'*r
        theta2 = theta - alpha.*grad
    end
    T = theta2
end
```

Ecuación normal

```
theta = X\y
```

Normalizado de atributos

```
N = size(X,1)
mu = mean(X(:,2:end))
sigma = std(X(:,2:end))
Xp = ones(nrows, ncols)
Xp(:,2:end) = (X(:,2:end) - repmat(mu,N,1)) ./ repmat(sigma,N,1)
```

Figura 1: Estudio previo a la práctica.

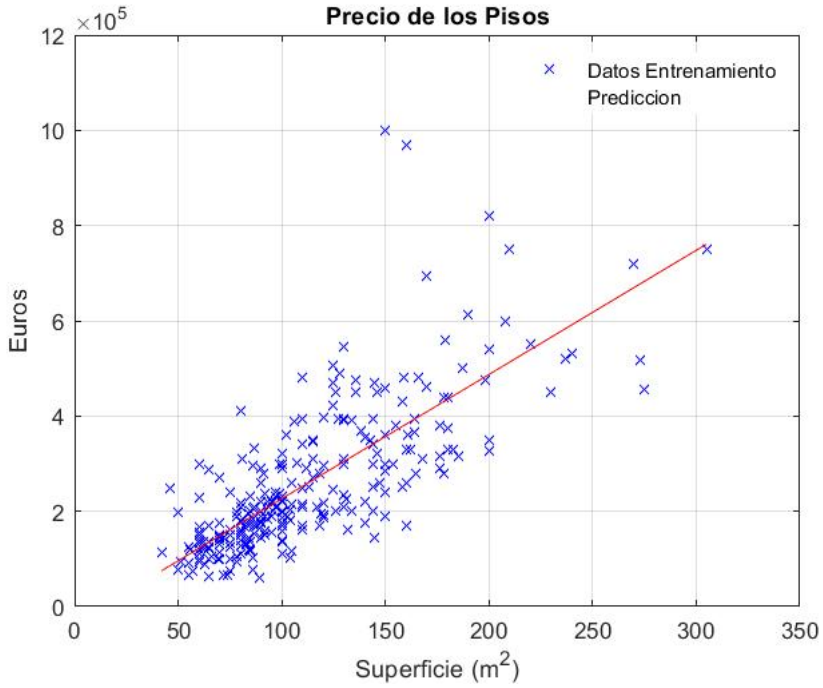

Figura 2: Recta de ajuste de predicción utilizando ecuación normal. Datos de entrenamiento.

3. Regresión monovariable utilizando la ecuación normal

En la predicción con regresión monovariable se ha utilizado el atributo de la superficie de los pisos. Inicialmente, ha sido necesario construir el vector de pesos que genera la ecuación normal. Para ello, he implementando una función (ecuación normal), que dada la entrada y la salida de los datos, te proporciona un vector de pesos.
Esta función se calcula de la siguiente manera:

$$T = X \backslash y; \tag{1}$$

Siendo T el vector de thetas a devolver.
Para realizar el entrenamiento, primero se han cargado los datos y obtenido los atributos. A continuación se ha formado el conjunto de datos de manera que se obtiene una matriz cuya primera columna contiene 1's y su segunda columna contiene la cantidad de metros cuadrados que tiene un piso. De la misma manera, se obtiene en el vector y el precio real del piso en euros.

Una vez obtenidos los datos de entrenamiento, se pasa a la función *ecuacionNormal* estos parámetros y se obtiene el vector de pesos. Con los pesos obtenidos se ha entrenado el modelo para obtener la predicción del precio de los pisos.

Se muestran los resultados obtenidos en una gráfica. Primero se ha dibujado en una gráfica los datos de entrenamiento, dibujados mediante puntos azules. A continuación, se ha dibujado la recta que predice el precio de los pisos, tal y como se puede ver en la figura 2.
Como se puede observar en la gráfica, la recta se aproxima bastante a los datos de entrenamiento en el extremo izquierdo, cuando la superficie del piso es pequeña. Sin embargo, cuanto mayor es la superficie del piso que se va a analizar, más imprecisa es esta predicción. Estas observaciones indican que puede haber más factores influyentes en el precio de un piso además de su superficie.

Otra medida para comprobar la corrección del algoritmo es mediante el error de predicción obtenido. Es decir, se calcula el vector de residuos para cada conjunto de datos. El vector de residuos se calcula de la siguiente manera:

$$r = X * theta - y \tag{2}$$

A continuación se pide comparar los residuos obtenidos con los puntos de entrenamiento y los de test. Al calcular el residuo de cada uno de estos conjuntos de datos (fórmula 2), se puede observar que, a priori no son comparables, ya que se obtienen vectores de residuos. Como la cantidad de datos de entrenamiento y la cantidad de datos de test no es la misma, no se pueden comparar ya que tienen distinta longitud. Por lo tanto, es necesario transformar los residuos a un número (magnitud) en la que sí sean comparables. Una idea intuitiva podría ser elegir la media aritmética como medida para comparar los errores. Sin embargo, como los errores pueden ser positivos, o negativos (puedes predecir que el precio de un piso es mayor de lo que en realidad es, o que el precio del piso es menor de lo que en realidad es) es necesario obtener una magnitud de errores en la que no se puedan compensar los errores positivos con los errores negativos. Podría usarse el valor absoluto como función para calcular los errores medios, pero, como ya se explicó en clase, esta función no es derivable, y por lo tanto, no es útil para los cálculos.
Por lo tanto, la mejor opción para calcular el error, es sumar sus cuadrados. Pero como los errores tienen unidades, utilizamos la función RMSE (3), para calcular la cantidad media de error en euros.

$$RMSE = \sqrt{\frac{r^T * r}{N}} \tag{3}$$

De esta manera, con el modelo ya entrenado, podemos calcular tanto el error con los datos de entrenamiento, como el error que se obtiene con los datos de test.

El error medio que se obtiene con los datos de entrenamiento es: $RMSEtrain = 1{,}0274e + 05$.
El error medio que se obtiene con los datos de test es: $RMSEtest = 8{,}0629e + 04$.

Como se puede observar, el error obtenido con los datos de test, es menor que el error obtenido con los datos de entrenamiento. Se puede observar como se produce subajuste, ya que el error en las muestras de entrenamiento es bastante alto, y el error con las muestras de test es algo menor. Esto es debido a que se tienen pocas muestras con las que entrenar el modelo, y también hay pocas muestras de test. Además, el modelo es muy sencillo, factor que también puede producir subajuste.

4. Regresión multivariable utilizando ecuación normal

En este ejercicio se calcula una predicción para el precio de los pisos utilizando regresión multivariable en función de la superficie de los pisos y el número de habitaciones que tiene haciendo uso de la ecuación normal (formula 1).

Primero se han cargado los datos de nuevo, pero esta vez, se ha añadido también el número de habitaciones que tiene cada piso. De esta manera, se obtiene una matriz de datos iniciales (X) que contiene en la primera columna 1's, en la segunda columna, se encuentra la superficie de los pisos y en la tercera la cantidad de habitaciones. Además, se dispone en el vector y el precio real de los pisos.

En este caso, como se tienen dos atributos que están en distinta magnitud se han normalizado los datos para obtener una mayor precisión aunque no es estrictamente necesario. Para la normalización y des-normalización se han creado dos funciones. A la primera función se le pasa el conjunto de datos y devuelve, los datos normalizados, el vector que contiene la media de cada uno de los atributos y el vector que contiene la desviación típica. Cabe destacar que, el primer atributo de los datos (la primera columna) contiene 1's y no debería influir en la normalización, por lo tanto, este atributo no se modifica.

```
function [Xp, mu, sigma] = normalizar(X)
    Xp = X;
    N = size(X,1);
    mu = mean(X(:,2:end));
    sigma = std(X(:,2:end));
    Xp(:,2:end) = (X(:,2:end)- repmat(mu,N,1)) ./ repmat(sigma,N,1);
end
```

Para la desnormalización también se ha creado una función, a la que se le pasa como parámetros el vector de pesos, la media calculada en la normalización y la desviación típica que se había obtenido. La función devuelve un nuevo vector de pesos.

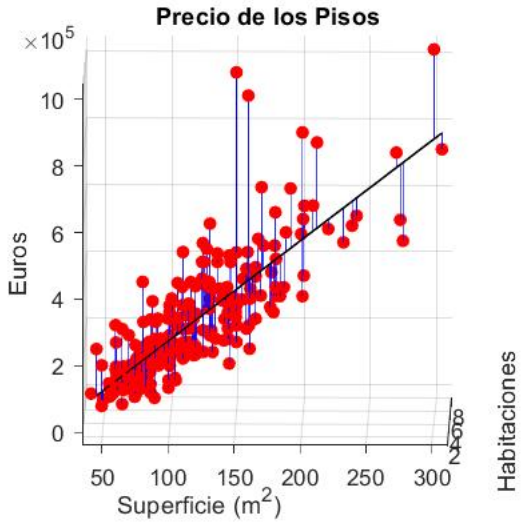

Figura 3: Plano de ajuste de predicción utilizando ecuación normal con regresión multivariable. Datos de entrenamiento. Perspectiva superficie.

```
function [ wdes ] = desnormalizar( w, mu, sig)
    wdes = w(2:end)./sig;
    wdes = [w(1)-(mu*des);wdes];
end
```

Estas funciones se utilizan en los algoritmos posteriores cuándo se considera necesario normalizar los atributos.

A continuación, para entrenar el modelo, primero se normalizan los datos con la función indicada anteriormente (4). Luego se aplica la ecuación normal (1) para obtener el vector de pesos, con los datos normalizados, y por último se desnormaliza el vector de pesos con la función indicada anteriormente (4). Una vez se obtiene el vector de pesos, se calcula la predicción de los datos de salida.
Para valorar el modelo, se muestra en una gráfica, los datos de entrenamiento con el plano de la predicción calculada (figura 3 y figura 4).

Como se pude observar en la gráfica (3) se ajusta considerablemente bien la predicción realizada con respecto a los resultados reales. Cabe destacar que, igual que pasaba con el algoritmo anterior, se ajusta mejor en pisos pequeños que en pisos grandes. En la segunda gráfica (4), se puede observar como los pisos grandes ven influenciado su precio según la cantidad de habitaciones que tienen. De la misma manera, se puede observar como, cuanto mayor es la superficie del piso, tiene más habitaciones y por consecuente, mayor es su precio.

A continuación, se comparan los residuos obtenidos con los datos de test y entrenamiento. Como ya se ha explicado antes, para comparar los residuos se utiliza el RMSE (3).

El error medio que se obtiene con los datos de entrenamiento es: $RMSEtrain = 1{,}0131e+05$.
El error medio que se obtiene con los datos de test es: $RMSEtest = 7{,}4741e+04$.

De la misma manera que con el algortimo anterior, se puede observar que el error obtenido con los datos de test es menor que el error obtenido con los datos de entrenamiento. En este caso, se puede considerar que hay otros atributos desconocidos influyentes en el modelo, o, que por el contrario, al haber pocas muestras de entrenamiento del modelo, se produce subajuste.
Si se comparan los resultados de error obtenidos en este modelo, con los resultados de error obtenidos

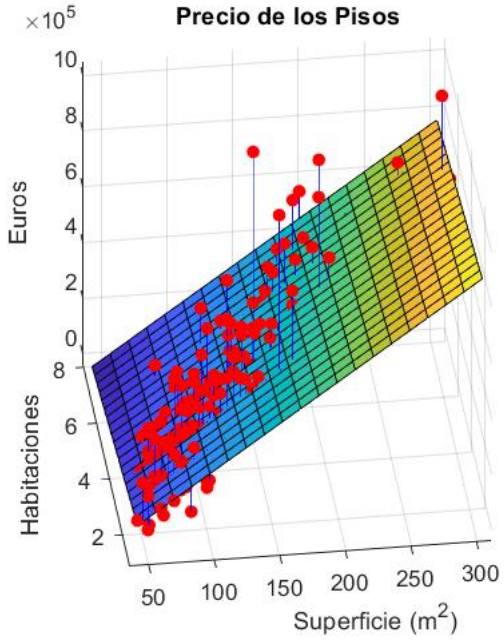

Figura 4: Plano de ajuste de predicción utilizando ecuación normal con regresión multivariable. Datos de entrenamiento. Perspectiva del número de habitaciones.

en el apartado anterior, se puede ver como tanto el error obtenido con los datos de entrenamiento, como el error obtenido con los datos de test es menor. Por lo tanto, para este conjunto de datos, la regresión multivariable con los atributos de superficie y número de habitaciones es mejor que la regresión monovariable con el atributo de superficie.

Por último, se pide calcular cuánto costaría un piso de $100m^2$ con 2, 3, 4 o 5 dormitorios y cuál de los dos modelos es mejor.
Para el modelo de regresión monovariable, solo será necesario tener en cuenta la superficie del piso ya que el número de habitaciones no influye en su predicción. Por lo tanto, se calcula el precio de un piso de $100m^2$ independientemente de la cantidad de habitaciones que tenga. El resultado obtenido es que dicho piso cuesta 226 510€. A continuación, se ha generado una matriz de entrada para simular pisos de $100m^2$ que tengan 2, 3, 4 y 5 habitaciones. La salida para estos pisos es de 253 040€, 234 180€, 215 330€y 196 480€respectivamente.

Como conclusión, se puede observar que el segundo modelo es mejor que el primero, ya que tiene en consideración que el precio del piso puede variar dependiendo de la cantidad de habitaciones que tenga ya que en la realidad, es así. Llama la atención como en el segundo modelo, el precio de los pisos disminuye contra más habitaciones tiene. Esto puede ser debido a que posiblemente no se haya considerado que un piso de $100m^2$ pueda tener 2 habitaciones y por consecuente, el modelo no predice bien en estos casos.

5. Regresión monovariable utilizando descenso de gradiente

En este apartado se pide realizar una regresión monovariable utilizando el descenso de gradiente. Para ello, primero se ha construido una función, similar a la desarrollada en el apartado previo para calcular los pesos mediante el descenso de gradiente.

La función construida para calcular el descenso de gradiente tiene como parámetros el conjunto de datos, un vector de pesos inicial, la salida de los parámetros y un factor de aprendizaje. Esta función devuelve como salida el vector de pesos calculado y el vector de coste en cada iteración (contiene en la componente i-esima el coste en la iteración i).
Se ha decidido, iterar calculando el descenso de gradiente hasta que la variación media de las thetas sea menor que 0.001 ya que tras realizar sucesivas pruebas, se ha podido observar que con este valor se

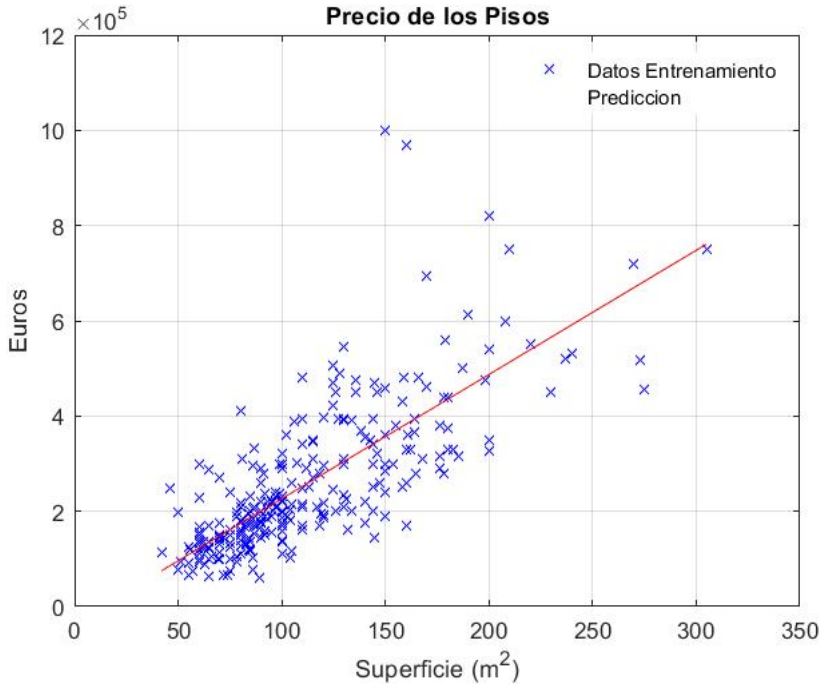

Figura 5: Recta de ajuste de predicción utilizando descenso de gradiente con regresión monovariable.

obtiene suficiente precisión y no realiza una cantidad excesiva de iteraciones.

La función creada es la siguiente:

```
function [T, J] = descensoDeGradiente(X, theta, y, alpha)
    r = X*theta - y;
    grad = X' * r;
    theta2 = theta - alpha .* grad;
    J = (1/2)*sum(r.^2);
    while (mean(abs(theta-theta2))>0.001)
        theta = theta2;
        r = X * theta - y;
        J = [J (1/2)*sum(r.^2)];
        grad = X' * r;
        theta2 = theta - alpha .* grad;
    end
    T = theta2;
end
```

Esta función realiza sucesivas iteraciones calculando el descenso de gradiente hasta encontrar un valor de theta que converja.

Para realizar la regresión, primero se han normalizado los datos utilizando la función explicada en el apartado anterior (función 4), luego se ha calculado el vector de pesos mediante el algoritmo de descenso de gradiente indicado (función 5). Para calcular el descenso de gradiente se le ha utilizado como factor de aprendizaje $alpha = 0{,}0001(10^{-4})$ ya que se ha obtenido una mayor precisión que utilizando $alpa = 10^{-3}$ y se ha obtenido la misma precisión que utilizando $alpa = 10^{-5}$ pero realiza muchas menos iteraciones. Por lo tanto, se ha decidido que el valor óptimo para el factor de aprendizaje es $alpa = 10^{-4}$. Posteriormente se han desnormalizado los datos obteniendo el vector de thetas real.

Tal y como se puede ver en la gráfica (5), la recta de ajuste resultante es muy similar a la obtenida en el apartado de regresión monovariable con la ecuación normal. Esto se debe a que se obtiene mucha precisión con este algoritmo.
En la siguiente gráfica (6) se muestra la evolución del coste del algoritmo en cada iteración. Se puede

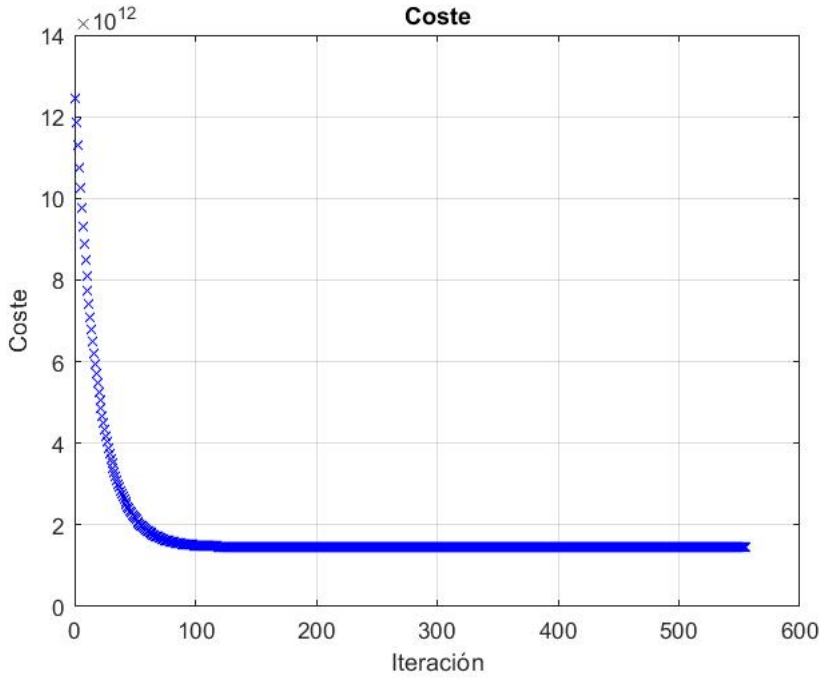

Figura 6: Curva de coste. Algoritmo de descenso de gradiente (regresión monovariable).

observar como el coste desciende rápidamente en las primeras iteraciones y, con el resto de iteraciones se reduce hasta que converge en un valor, demostrando así la corrección del algoritmo y de los parámetros elegidos.

Para comparar este algoritmo con la regresión monovariable utilizando ecuación normal se utilizar el RMSE obtenido con ambos algoritmos.

El error medio que se obtiene con los datos de entrenamiento es: $RMSEtrain = 1{,}0274e + 05$.
El error medio que se obtiene con los datos de test es: $RMSEtest = 8{,}0629e + 04$.

Como se puede observar se obtienen exactamente los mismos errores que en con el algoritmo de regresión monovariable utilizando la ecuación normal. Esto es debido a que el algoritmo de descenso de gradiente se aproxima con mucha precisión al valor de theta real. Además, al haber asignado un valor del factor de aprendizaje bastante pequeño y haber normalizado previamente los datos, se obtiene un vector de pesos muy preciso, obteniendo de este modo, unos resultados muy similares al apartado 3.

6. Regresión multivariable utilizando descenso de gradiente

En este ejercicio se calcula una predicción para el precio de los pisos utilizando regresión multivariable utilizando descenso de gradiente. Los datos de entrenamiento contienen como atributos la superficie de los pisos y la cantidad de habitaciones que tienen.

Primero se han cargado los datos y se ha formado el conjunto de datos de entrada con los atributos especificados.

Para construir el modelo, se normalizan los datos, igual que se ha hecho en los otros algoritmos. Esto se realiza mediante la función (4). A continuación, se calcula el vector de pesos mediante el algoritmo de descenso de gradiente (5) y, posteriormente de desnormalizan los pesos. Para realizar las iteraciones con el algoritmo de descenso de gradiente se han utilizado los mismos parámetros que en el apartado anterior y por los mismos motivos. La función de coste obtenida en este caso es la adjuntada en la figura 7.

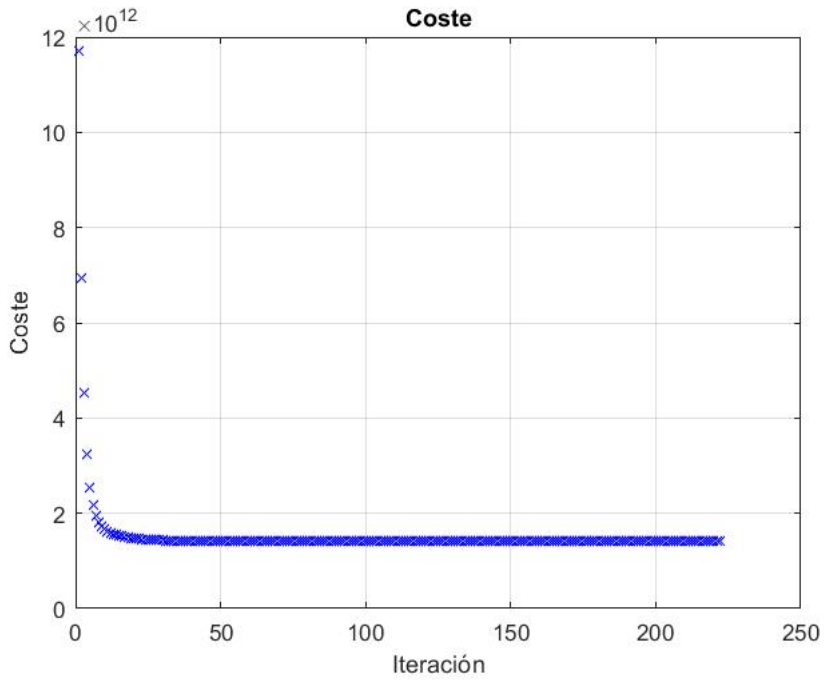

Figura 7: Curva de coste. Algoritmo de descenso de gradiente (regresión multivariable).

Como se puede observar en la gráfica, el coste converge. Esto demuestra la corrección del algoritmo y del vector de pesos calculado.
También se muestran los resultados de la predicción mostrados en el plano calculado, junto con los datos de entrenamiento. Estos resultados se pueden observar en la gráfica 8. Como conclusión, cabe destacar que esta predicción en muy parecida a la calculada en la sección 4 (regresión multivariable utilizando ecuación normal). Esto es debido a, como ya se ha explicado antes, la alta precisión del algoritmo.

Como ya se ha realizado con los otros algoritmos, se puede comparar el error de los algoritmos para valorar su eficacia.

El error medio que se obtiene con los datos de entrenamiento es: $RMSEtrain = 1{,}0131e+05$.
El error medio que se obtiene con los datos de test es: $RMSEtest = 7{,}4741e+04$.

Como podemos ver, se ha obtenido exactamente el mismo error que se obtiene para el algoritmo de regresión lineal multivariable con ecuación normal. Por lo tanto, en este modelo también se está causando subajuste. Los motivos del subajuste son exactamente los mismos que en el algoritmo de la sección 4.

7. Regresión robusta con el coste de Huber

En este apartado se pide calcular una predicción para el precio de los pisos utilizando regresión robusta con el coste de Huber.

Para ello, se han creado dos funciones. La primera de ellas (función 7) devuelve el coste y el gradiente dados un vector de pesos, los datos de entrada, los datos de salida reales y el parámetro delta, que indica el umbral a partir del que comienzan a considerarse datos espurios. Es decir, delta es el parámetro del error admitido para el que se asigna un peso normal a los datos. Si un dato contiene un error mayor que delta, se le asigna menor peso en la predicción. De esta manera, los datos espurios tienen un menor peso en la predicción y por consecuente influyen menos.

```
function [J,grad,Hess] = CosteHuber2(theta,X,y,d)
```

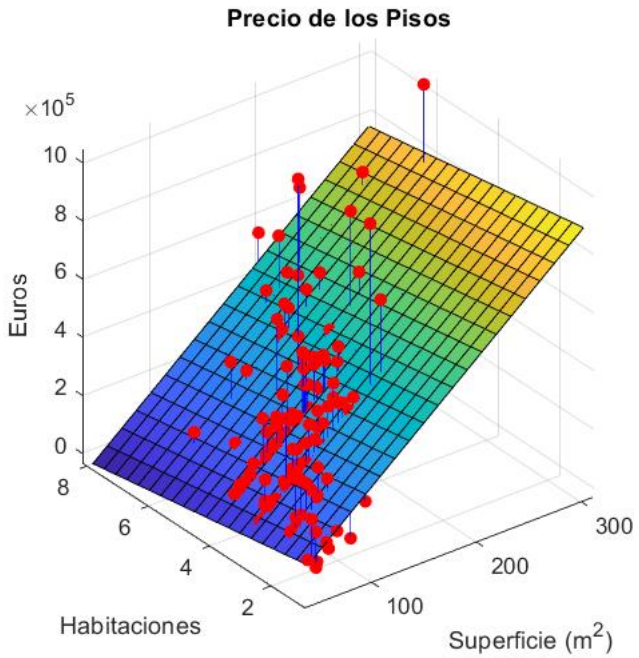

Figura 8: Plano de ajuste de predicción utilizando descenso de gradiente con regresión multivariable.

```
    r = X*theta-y;
    good = abs(r) <= d;
    J = (1/2)*sum(r(good).^2) + ...
        d*sum(abs(r(~good))) - (1/2)*sum(~good)*d^2;
    if nargout > 1
        grad = X(good,:)'*r(good) + ...
         d*X(~good,:)'*sign(r(~good));
    end
    if nargout > 2
        Hess = X(good,:)'*X(good,:);
    end
end
```

Esta función también te permite calcular el Hessiano con el coste de Huber si fuera necesario.
La otra función construida calcula el vector de pesos adecuado utilizando descenso de gradiente. Para ello, se ha utilizado una función similar a la utilizada en los apartados anteriores para calcular el descenso de gradiente (5), pero esta vez el gradiente y el coste lo calcula la función del Coste de Huber (7). Se realizan iteraciones de descenso hasta que el coste converja.
La función construida es la adjuntada en la función 7. En este caso se ha decidido realizar iteraciones de descenso de gradiente hasta que la variación media del vector de thetas sea menor que 0.001. Esto se debe a que, tras realizar varias pruebas, se ha comprobado que realizar más iteraciones de descenso no mejora la precisión del modelo.

```
function [T, J] = CosteHuber(X, y, theta, alpha, d)
    [J, grad] = CosteHuber2(theta,X,y,d);
    theta2 = theta - alpha .* grad;
    while (mean(abs(theta-theta2))>0.001)
        theta = theta2;
        [J2, grad] = CosteHuber2(theta,X,y,d);
        J = [J J2];
        theta2 = theta - alpha .* grad;
    end
    T = theta2;
end
```

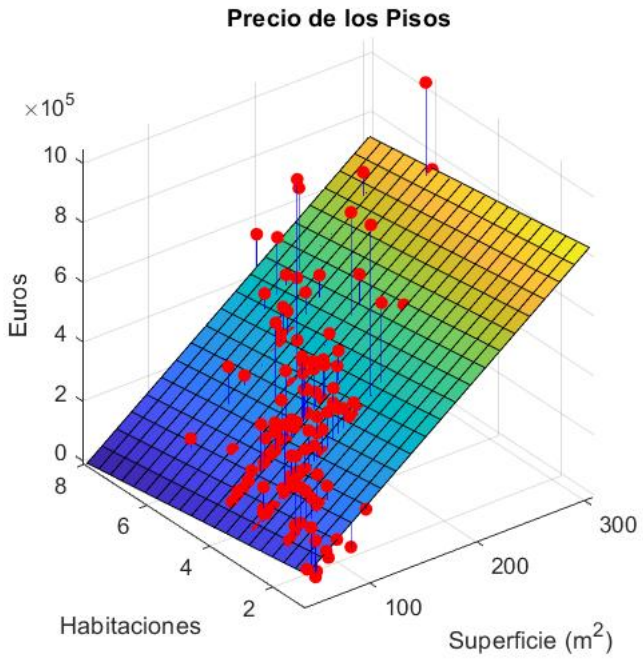

Figura 9: Plano de ajuste de predicción utilizando regresión robusta con el coste de Huber.

Para construir el modelo se han cargado los datos, se han normalizado los datos con la función de normalización (4). A continuación se ha calculado el vector de pesos con la función de $CosteHuber$ (7) y posteriormente se han desnormalizado los datos con la función de desnormalización (4).
Los parámetros elegidos han sido $alpha = 0{,}001$ como factor de aprendizaje. Se ha elegido este valor ya que, se ha probado con distintos valores y este se ha considerado el más adecuado. Al ejecutar el algoritmo con $alpha = 0{,}01$ el descenso se realiza muy rápido y por tanto oscila. Al probar con el valor $alpha = 0{,}0001$ se ha comprobado que desciende muy lentamente y, por lo tanto se realizan iteraciones innecesarias. Por lo tanto, se considera $alpa = 0{,}001$ valor óptimo para el factor de aprendizaje en este caso.
En cuanto al valor *delta* (umbral de error para considerar datos como espurios), se ha utilizado $delta = 1{,}5 * 10^5$ ya que como se ha podido observar en los modelos anteriores el valor medio del error es aproximadamente $1{,}5 * 10^5$ y por consecuente, se ha considerado que este valor sería correcto. También se ha probado a ejecutar el algoritmo con otros parámetros de delta y tal como pasa con el parámetro alpha, solo se realizan iteraciones innecesarias al aumentar este parámetro.

A continuación, se muestra en una gráfica el plano obtenido en la predicción junto con los datos de entrenamiento. Como se puede observar, esta gráfica (figura 9) aunque es muy parecida a las obtenidas en algoritmos anteriores, se ajusta mejor a los datos aparentemente.

Para comprobar el correcto funcionamiento del algoritmo, al igual que en los algoritmos anteriores se muestran distintas medidas.Se muestra en la figura 10 como converge el coste en las sucesivas iteraciones del bucle. Esto demuestra la corrección del algoritmo.

También se puede mostrar el error medio obtenido con el modelo para los datos de entrenamiento y test en este modelo.

El error medio que se obtiene con los datos de entrenamiento es: $RMSEtrain = 1{,}0188e + 05$.
El error medio que se obtiene con los datos de test es: $RMSEtest = 7{,}1414e + 04$.

Si se compara con los errores obtenidos en los otros algoritmos, se puede observar que, con los datos de entrenamiento se obtiene un error mayor que con los datos de test. Luego, se produce subajuste de nuevo.
La regresión robusta, tiene un error algo mayor que la regresión multivariable (tanto con ecuación nor-

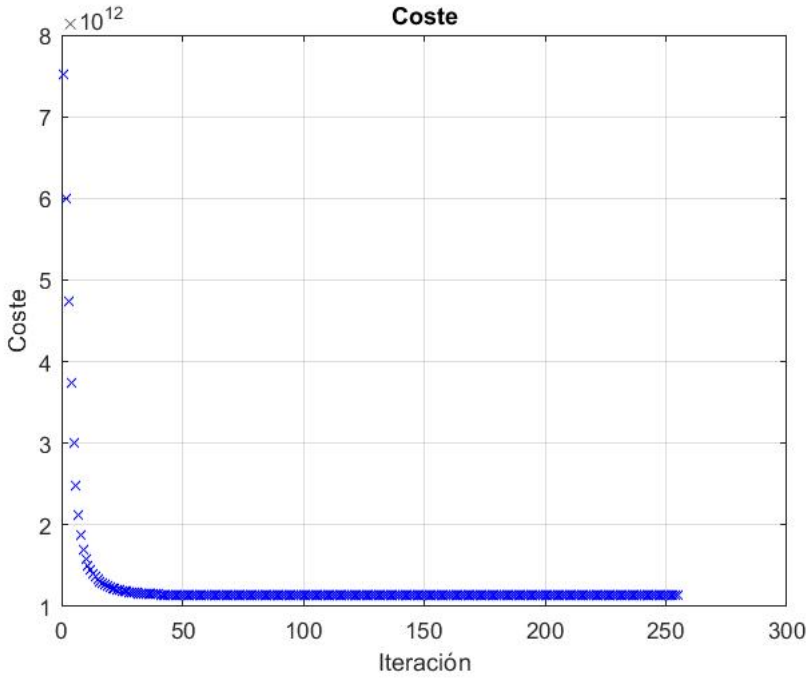

Figura 10: Curva de coste. Algoritmo de regresión robusta con el coste de Huber (descenso de gradiente multivariable).

mal como con descenso de gradiente), sin embargo, tiene un error menor que la regresión monovariable (tanto con ecuación normal como con descenso de gradiente). Ahora, si comparamos el error obtenido con los datos de test, se observa que se obtiene un error menor que el obtenido con todos los algoritmos anteriores. Por lo tanto, se puede confirmar que este modelo es mejor que todos los anteriores.