Práctica 2

08/03/2022

César Borja Moreno 800675

Aprendizaje Automático

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Índice

Trabajo previo 3
Selección del grado del polinomio para la antigüedad del coche. 3
Selección del grado del polinomio para los kilómetros. 4
Selección del grado del polinomio para la potencia. 5
Regularización 6

Anexo 1 9

1. Trabajo previo

El trabajo previo consiste en programar el algoritmo de k-fold cross-validation para elegir una regresión polinómica o el valor del parámetro de regularización. El algoritmo propuesto inicialmente es el siguiente:

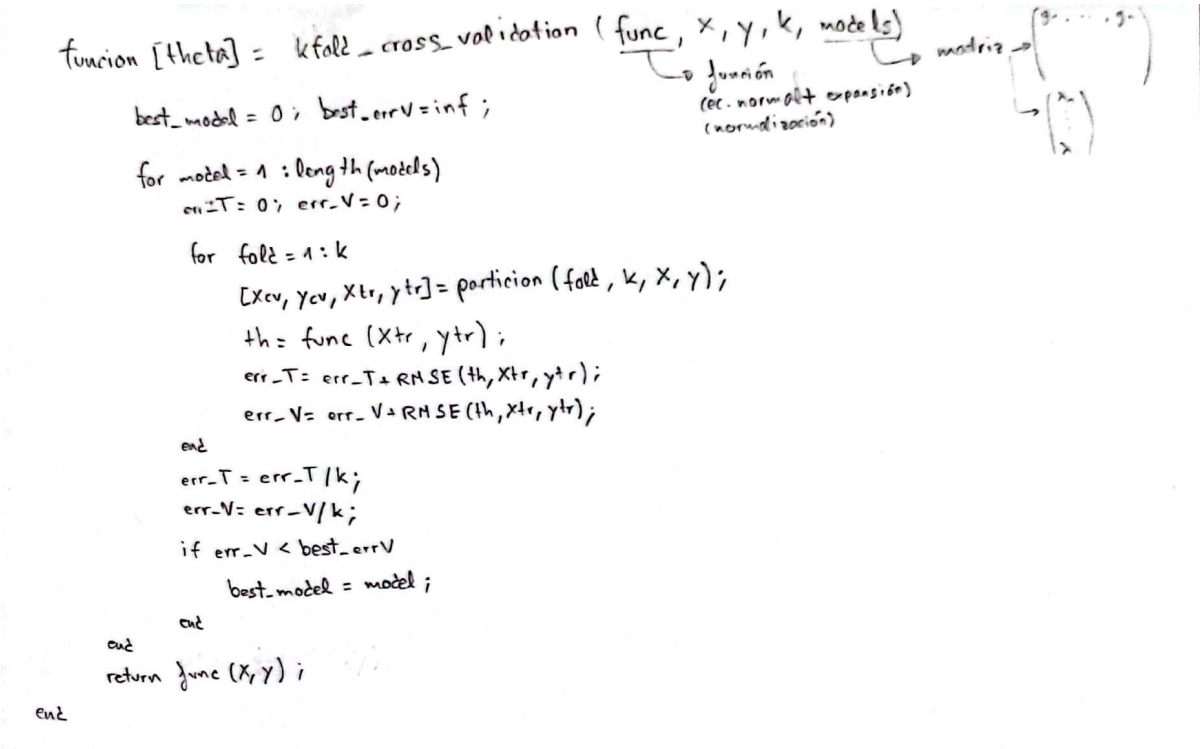

Figura 1. Trabajo previo

Finalmente, el algoritmo implementado y corregido se muestra en el Anexo 1.

2. Selección del grado del polinomio para la antigüedad del coche.

Se pide utilizar el algoritmo de k-fold propuesto en el trabajo previo para obtener el grado del polinomio para la antigüedad del coche (entre 1 y 10) manteniendo fijos los kilómetros y la potencia a 1. Las curvas de los errores RMSE de entrenamiento y validación se muestran en la Figura 2.

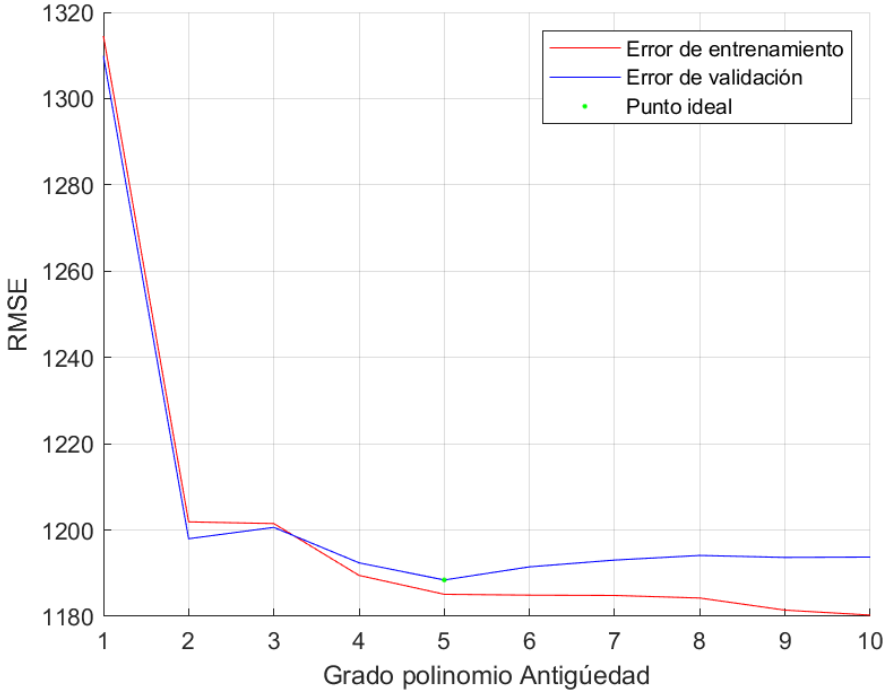

Figura 2. Curvas de evolución de los errores RMSE de entrenamiento y validación según el grado del polinomio para la antigüedad del coche

Se busca obtener el grado del polinomio que minimice el RMSE de validación para elegir el mejor modelo de cara a evaluar los datos de test en el futuro.

Como se puede observar el polinomio que proporciona el menor RMSE de validación para la antigüedad es el polinomio de grado 5. A partir del grado 5 se aprecia sobreajuste, aunque no muy elevado.

3. Selección del grado del polinomio para los kilómetros.

Una vez obtenido el grado del polinomio para la antigüedad se pide conseguir el mejor grado del polinomio para los kilómetros (entre 1 y 10), manteniendo el grado de la potencia a 1 y la antigüedad a 5. En la Figura 3 se muestran las curvas de los errores RMSE de entrenamiento y validación.

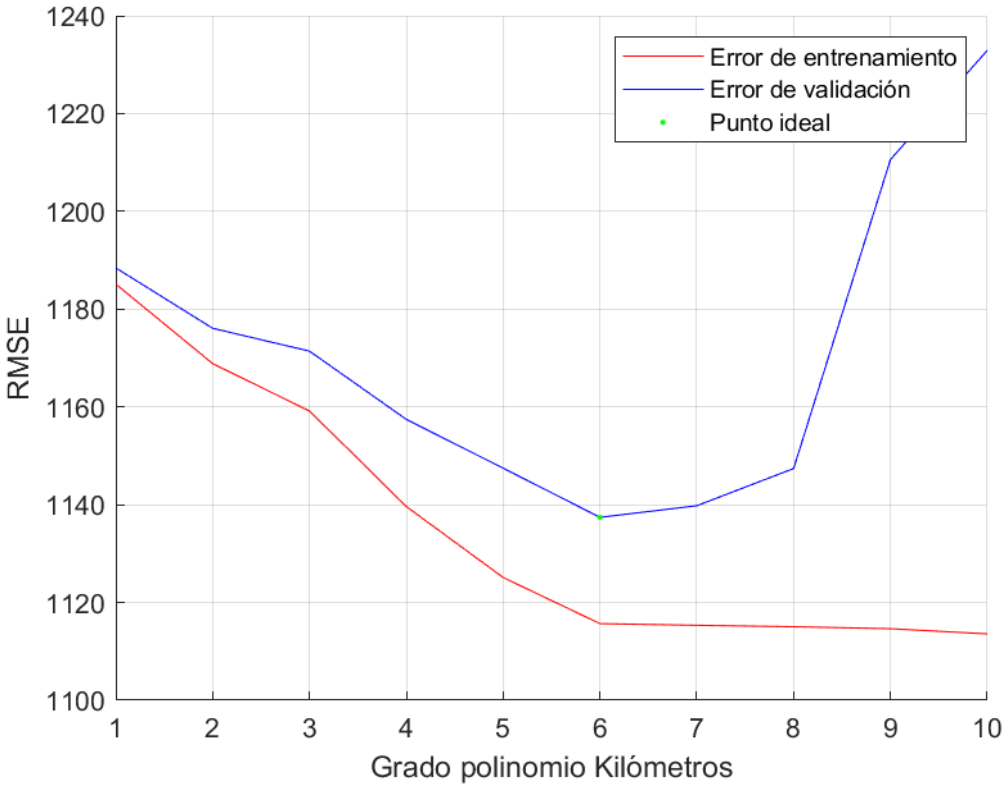

Figura 3. Curvas de evolución de los errores RMSE de entrenamiento y validación según el grado del polinomio para los kilómetros del coche manteniendo grado 5 para la antigüedad.

Se puede observar que el RMSE de validación más bajo se consigue con el polinomio de grado 6. En este caso sí que se aprecia un claro sobreajuste para los polinomios de grado 8, 9 y 10.

4. Selección del grado del polinomio para la potencia.

Manteniendo fijos los mejores grados del polinomio para la antigüedad y los kilómetros, se busca por último el mejor grado del polinomio para la potencia del coche. Las curvas de evolución del RMSE de entrenamiento y validación se pueden ver en la Figura 4.

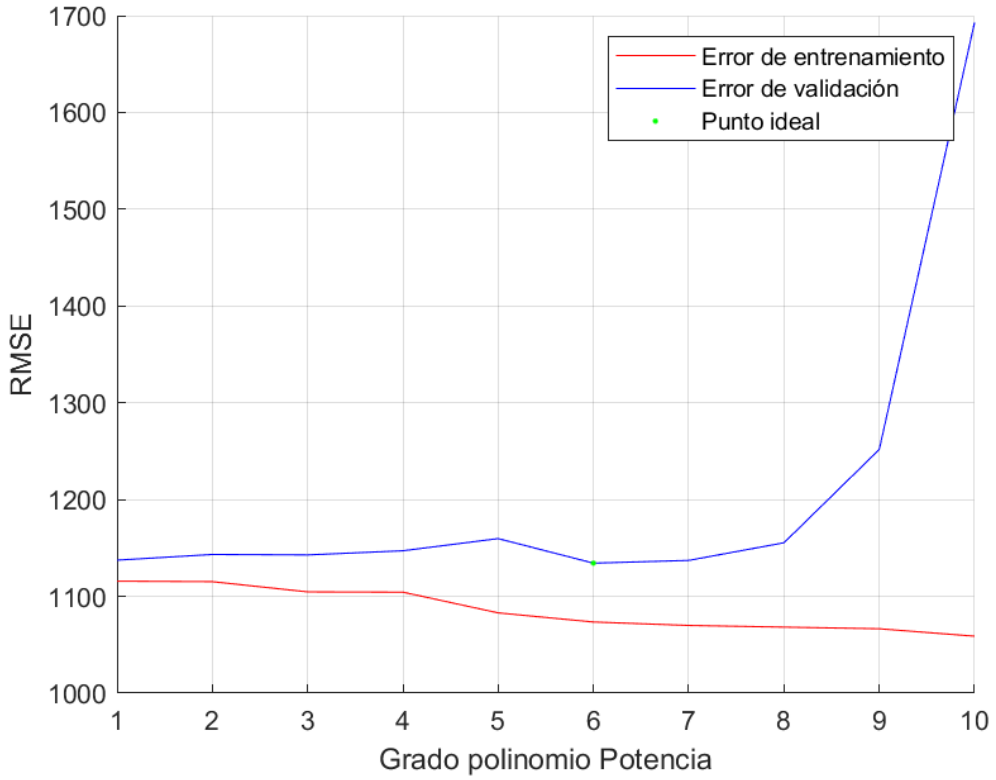

Figura 4. Curvas de evolución de los errores RMSE de entrenamiento y validación según el grado del polinomio para la potencia del coche manteniendo grado 5 para la antigüedad y grado 6 para los kilómetros.

Se puede observar que de nuevo aparece sobreajuste para los grados 8, 9 y 10. El mejor grado para la potencia es 6, tal y como muestra el punto ideal la gráfica.

Como resultado de los últimos 3 apartados, se puede concluir en que el mejor modelo polinómico para predecir el precio de un coche de segunda mano en función de su antigüedad, sus kilómetros y su potencia es aquel cuyo polinomio utiliza grados 5, 6 y 6 para la antigüedad, kilómetros y potencia respectivamente.

Ahora, utilizando el mejor modelo encontrado (modelo 1), se ha entrenado con todos los datos de entrenamiento (sin utilizar datos de validación) y se ha calculado el error RMSE con los datos de test. Esto se hace de la siguiente forma en MATLAB:

```
%entrenar con todos los datos
X_exp_train= expandir(Xdatos,best_grades); %mejor modelo encontrado
[Xn,mu,sig] = normalizar(X_exp_train);
th = Xn \ ydatos;
[th] = desnormalizar(th,mu,sig);

%calcular RMSE con datos de test
X_exp_test= expandir(Xtest,best_grades);
err_test = RMSE(th,X_exp_test,ytest);
```

El RMSE de los datos de test es:

$$RMSEmodelo1 = 1.0093e{+}03$$

5. Regularización

Por último se pide realizar el ajuste de un polinomio de grado 10 para los tres parámetros, utilizando regularización.

Los valores de lambda que se han obtenido de la siguiente manera:

```
i = 0.00000001;
lambda = [];
while i < 0.00001
    lambda = [lambda i];
    i = i .* 2;
end
```

No se ha elegido aumentar lambda de forma logarítmica ya que la variación del error de validación que se quiere observar ocurre en un rango de valores de lambda pequeño. Por tanto, para favorecer la visualización de las curvas RMSE, se aumenta el factor de regularización multiplicando por 2.

En la Figura 5 se muestran las curvas de evolución del RMSE de entrenamiento y validación.

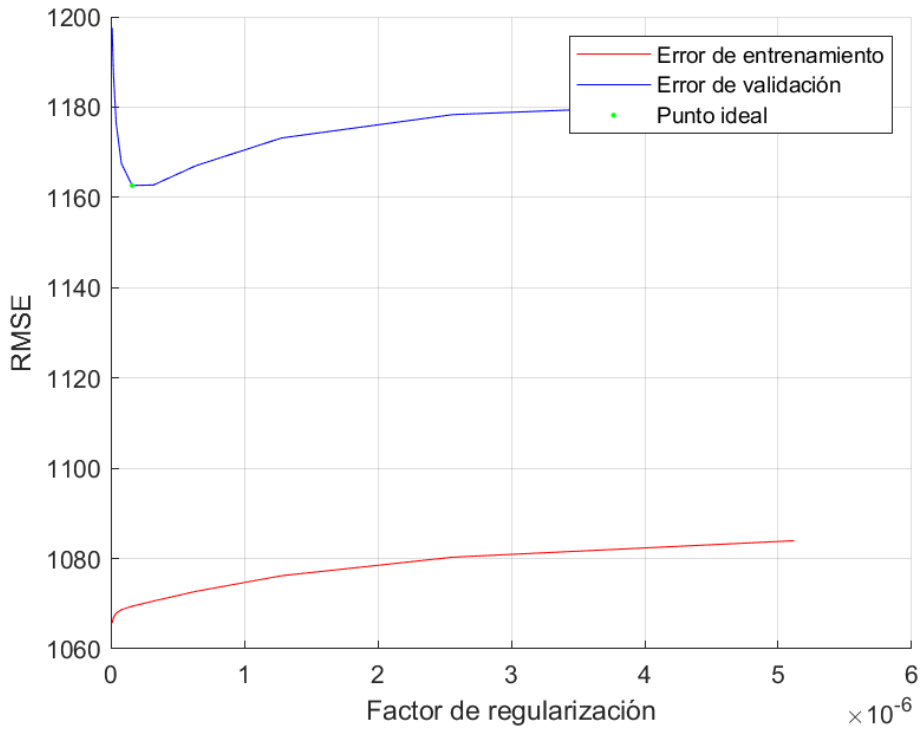

Figura 5. Curvas de evolución de los errores RMSE de entrenamiento y validación en función del factor de regularización λ.

Como se ha visto en los apartados 2, 3 y 4, utilizar un polinomio de grado 10 para los tres atributos va a producir sobreajuste (modelo demasiado complejo), y es por esto que se aplica la técnica de regularización. Sin embargo, observando las curvas de evolución del RMSE de entrenamiento y validación podemos apreciar como con valores muy pequeños de λ el impacto de la regularización es casi inexistente, presentando sobreajuste (el error de entrenamiento disminuye mucho mientras que el de validación aumenta drásticamente).

Por el contrario, usando valores grandes, tanto el RMSE de entrenamiento como el de validación crecen, es decir, se produce sub-ajuste. Esto ocurre porque estamos simplificando demasiado el modelo.

El valor óptimo para λ es aquel que minimiza el error de validación y que viene representado en la gráfica como el punto ideal ($\lambda = 0,016e\text{-}5$).

Una vez obtenido el valor óptimo para λ (modelo 2), ha sido entrenado con todos los datos de entrenamiento (sin utilizar datos de validación) y se ha calculado el error RMSE con los datos de test. Esto se hace de la siguiente forma en MATLAB:

```
%entrenar con todos los datos
X_exp_train= expandir(Xdatos,[10 10 10]);
[Xn,mu,sig] = normalizar(X_exp_train);
H = Xn'*Xn + lambda(best_model)*diag([0 ones(1,size(Xn,2)-1)]);
th = H \ (Xn'*ydatos);
[th] = desnormalizar(th,mu,sig);

%calcular RMSE con datos de test
X_exp_test = expandir(Xtest,[10 10 10]);
err_test = RMSE(th,X_exp_test,ytest);
```

El RMSE obtenido con los datos de test es:

$$RMSEmodelo2 = 1.0082e\text{+}03$$

¿Cuál de los dos modelos es mejor?

Para comparar el modelo en el que se seleccionan los grados del polinomio (modelo 1) con el modelo regularizado (modelo 2), se compara el RMSE de los datos de test:

$$RMSEmodelo2 = 1.0082e\text{+}03 < RMSEmodelo1 = 1.0093e + 03$$

Dado que el menor error se consigue con el modelo 2 (modelo que utiliza regularización), este es el mejor.

Anexo 1

```
function [theta,best_model,RMSEtr,RMSEcv] = kfold_cross_validation(reg, X, y, k, models)
    best_model = 0;
    best_errV = inf;
    RMSEtr = [];
    RMSEcv = [];
    [rows,~] = size(models);
    for model = 1:rows
        err_T = 0; err_V = 0;
        for fold = 1:k
            [Xcv,ycv,Xtr,ytr] = particion(fold,k,X,y);
            if(reg) %regularizacion
                [Xn,mu,sig] = normalizar(Xtr);
                H = Xn'*Xn + models(model)*diag([0 ones(1,size(Xn,2)-1)]);
                th = H \ (Xn'*ytr);
                [th] = desnormalizar(th,mu,sig);

                err_T = err_T + RMSE(th,Xtr,ytr);
                err_V = err_V + RMSE(th,Xcv,ycv);
            else
                Xtr_exp = expandir(Xtr,models(model,:));
                Xcv_exp = expandir(Xcv,models(model,:));
                [Xn,mu,sig] = normalizar(Xtr_exp);
                th = Xn \ ytr;
                [th] = desnormalizar(th,mu,sig);

                err_T = err_T + RMSE(th,Xtr_exp,ytr);
                err_V = err_V + RMSE(th,Xcv_exp,ycv);
            end
        end
        err_T = err_T / k;
        RMSEtr = [RMSEtr;err_T];
        err_V = err_V / k;
        RMSEcv = [RMSEcv;err_V];
        if err_V < best_errV
            best_errV = err_V;
            best_model = model;
        end
    end
    if(reg) %regularizacion
         H = Xtr'*Xtr + models(best_model)*diag([0 ones(1,size(Xtr,2)-1)]);
         theta = H \ (Xtr'*ytr);
    else
        Xtr_exp = expandir(Xtr,models(best_model,:));
        Xcv_exp = expandir(Xcv,models(best_model,:));
        theta = Xtr_exp \ ytr;
end
```