Elección de Hiperparámetros

¡Comencemos!

Competencias

- Conocer la mecánica de regularización en los métodos Ridge, Lasso y Elastic-net.
- Utilizar los métodos de regularización para resolver problemas de dimensionalidad y mejora de desempeño predictivo.
- Implementar los métodos con la librería sklearn.

introducción

En esta sección conocerás acerca de la mecánica de regularización en los métodos Ridge, Lasso y Elastic-net. Además, implementaremos los métodos con la librería sklearn.

¡Vamos con todo!

Búsqueda de hiperparámetros y early stopping

Como mencionamos, la elección de λ es definida por el investigador. Este tipo de valores los cuales nosotros debemos escoger se llaman Hiperparámetros. Cada modelo/agenda de entrenamiento tiene sus propios hiperparámetros los cuales deberemos saber interpretar si queremos sacar el mayor provecho a nuestro modelo.

El proceso de la búsqueda de hiperparámetros se puede ejemplificar con el siguiente diagrama:

- El primer paso es definir una serie de hiperparámetros candidatos a evaluar.
- Para cada uno de estos hiperparámetros candidatos, estimaremos un modelo mediante validación cruzada.
- Para cada paso dentro de la validación cruzada, evaluamos el hiperparámetro y estimamos su métrica. En este caso, estamos implementando una métrica de error cuadrático promedio.
- Al finalizar la validación cruzada, preservamos la métrica promediada.
- El mejor hiperparámetro será aquél que presente un mejor desempeño. En este diagrama, cuando $\lambda = 0.05$ el MSE de validación cruzada será el mejor.

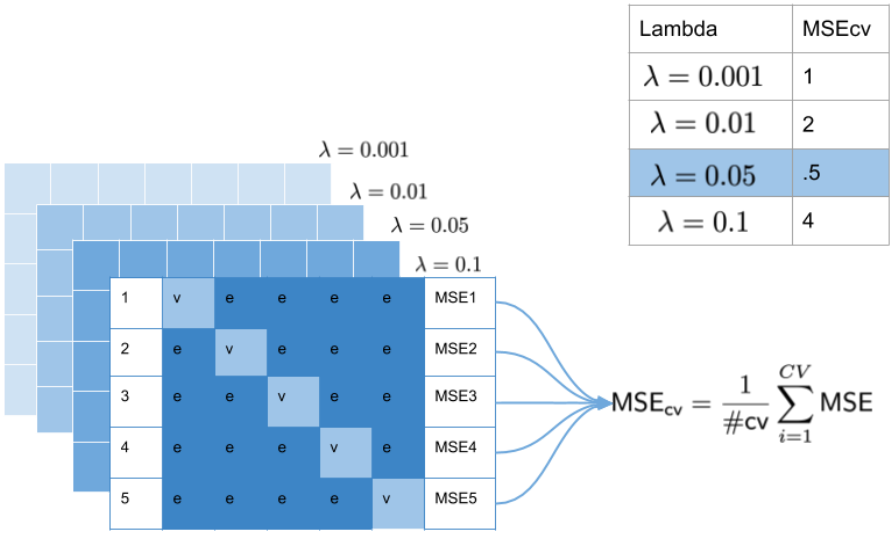

Imagen 1. Diagrama de hiper parámetros.
Fuente: Desafío Latam.

Si bien no existe una regla clara para escoger hiperparámetros (es un problema complejo por lo mismo), una técnica usual para explorar este tipo de problemas consiste en detener el entrenamiento/búsqueda de hiperparámetros óptimos cuando veamos algún tipo de convergencia o mínimo local. A esta idea de detener el entrenamiento antes de tiempo, se le conoce más formalmente como Early Stopping.

Puesto que entrenar un modelo de regresión lineal con una matriz de datos como la que tenemos es barato computacionalmente, nos daremos el lujo de evaluar varios valores para y elegiremos el que entrega menor error de validación, exploraremos el rango de valores de

$$\lambda \in [10^7, 10^3]$$

Para implementar un modelo con regularización `Ridge`, importamos las clases `Ridge` y `RidgeCV` dentro de `sklearn.linear_model`. Un punto de confusión que surge en la implementación de los métodos de regularización en sklearn es que el parámetro de penalización λ está implementado con el nombre de `alpha`.

```
from sklearn.linear_model import Ridge, RidgeCV
from sklearn.metrics import r2_score, mean_squared_error

names_regressors = X_train.columns # guardamos los nombres de los
atributos
alphas = np.logspace(0, 7, base = 10) # generamos un vector con los
valores de la norma
coefs_ridge = [] #lista para guardar parámetros
cv_err_ridge = [] #lista para guardar parámetros
model_ridge = Ridge(fit_intercept = True) # instanciamos el modelo
tol = 0.1 # determinamos el umbral de tolerancia

# para cada valor en el vector
for a in alphas:
   # estimamos el modelo con éste
   model_ridge.set_params(alpha = a)
   model_ridge.fit(X_train, y_train)
   # guardamos el coeficiente estimado
   coefs_ridge.append(model_ridge.coef_)
   # generamos su estimado de validación cruzada
   dummy,cv_err_estimates = gfx.cv_error(X_train, y_train, k = 10,
method = 'ridge', alpha = a)
   cv_err_ridge.append(np.mean(cv_err_estimates)) # OJO: estamos
guardando la media del error de cv para cada alpha


for y_arr, label in zip(np.squeeze(coefs_ridge).T, names_regressors):
```

```
    plt.plot(alphas, y_arr, label = label)

plt.legend()
plt.xscale("log")
plt.title("Ridge Regression: coeficientes vs par. de regularización",
size = 14)
plt.xlabel('Lambda')
plt.ylabel('Coef. de la regresión')
plt.axis("tight")
plt.legend(loc="center left", bbox_to_anchor=(1, .5));
```

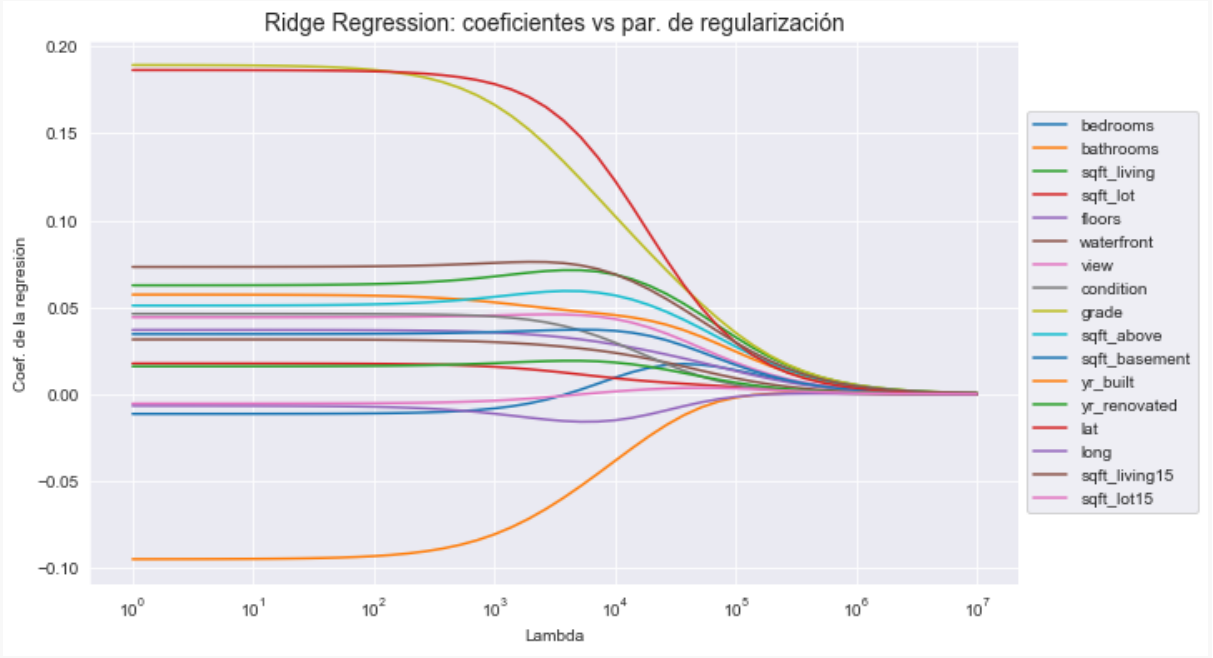

Imagen 2. Regresión Ridge.
Fuente: Desafío Latam.

A partir de este gráfico podemos ver algunas cosas interesantes:

- Tener un parámetro de regularización de 10^0 produce el mismo resultado en los coeficientes que un parámetro de 10^1 el último es muy débil y es casi inofensivo.
- En efecto, al tener un parámetro de regularización lo suficientemente grande, los coeficientes de algunos atributos se hacen cada vez más pequeños (cada vez se les penaliza más) hasta hacerse casi iguales a 0.
- Para un parámetro de regularización lo suficientemente grande, todos los coeficientes se hacen casi iguales a 0, por lo que no podemos excedernos mucho con el valor del mismo.

- Ridge se dice que es un método de contracción (shrinkage method) porque "contrae" los valores de los coeficientes de la regresión.

Veamos cómo se comporta el error de validación para cada λ.

```
plt.plot(alphas, np.sqrt(cv_err_ridge),"*-", color='dodgerblue')
plt.xscale("log")
plt.title("Ridge Regression: RMSE de Cross-Validation para cada
$\lambda$", fontsize = 14);
```

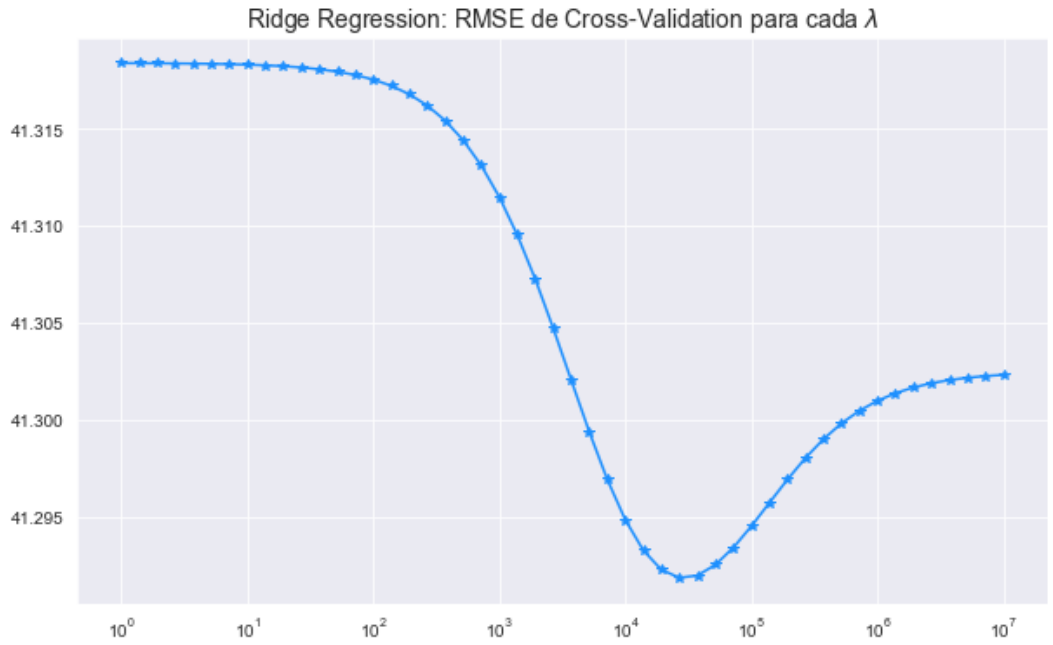

Imagen 3. Error de Validación.
Fuente: Desafío Latam.

Como mencionamos anteriormente, nuestro objetivo es encontrar el valor de λ mínimo, sin embargo, parecen haber varios valores que se ajustan a nuestro criterio. Aprovecharemos de introducir una función conveniente que implementa la librería `sklearn` llamada `RidgeCV`:

```
alphas_ = np.logspace(0, 7, base = 10)
ridge_cv = RidgeCV(cv = 10)
model_ridge = ridge_cv.fit(X_train, y_train)
```

El modelo elige automáticamente el valor de λ que minimiza el error de validación:

```
def report_regularization(model, X_test, y_test):
   print('Valor del parámetro de regularización:
{0}'.format(model.alpha_))
   print('Coeficientes finales: \n{0}'.format(model.coef_))
   y_hat =  model.predict(X_test)
   print('R-squared: {0}'.format(r2_score(y_test,y_hat)))
   print('Mean Squared Error: {0}'.format(mean_squared_error(y_test,
y_hat)))

report_regularization(ridge_cv, X_test, y_test)
```

```
Valor del parámetro de regularización: 10.0
Coeficientes finales:
[-0.01130049 0.05717332 0.06264874 0.01764058 0.03692392 0.03143451
0.04422176 0.04616988 0.18895082 0.05101663 0.03455404 -0.09467497
0.01602423 0.18624593 -0.0067097   0.07316506 -0.00536286]
R-squared: 0.7685417941024782
Mean Squared Error: 0.06231291960423319
```

Para finalizar nuestra aventura por Ridge, se debe mencionar que **Ridge no puede ocuparse como un método de selección de atributos, es decir, la cantidad de atributos** de una regresión lineal ordinaria es la misma que la cantidad de atributos que tenemos al hacer una regresión vía Ridge.

Digresión: Métricas de desempeño para problemas de regresión

Para medir el desempeño predictivo de un algoritmo que busque solucionar un problema de regresión, implementaremos alguna medida que cuantifique la distancia entre lo predicho y lo observado. Partamos por la métrica más conocida:

- **Error Cuadrático Promedio** (Mean Squared Error): Sintetiza la distancia entre lo predicho y lo observado. Es el criterio a optimizar cuando nuestro buscamos minimizar la suma de cuadrados residuales:

$$\text{MSE} = \frac{1}{n}\sum_{j=1}^{N}(y_j - \hat{y})^2$$

- El error cuadrático promedio permite diagnosticar si las predicciones realizadas por el modelo subestiman o sobreestiman el comportamiento, dado el signo. La magnitud de la métrica dependerá del rango de valores en el vector objetivo y se interpreta como el fallo promedio en la predicción de una observación cualquiera. Nos permite reportar la subestimación/sobrestimación del modelo respecto a la esperanza matemática del vector objetivo.

$$\mathbb{E}[y]$$

- **Raíz Cuadrada del Error Cuadrático Promedio** (Root Mean Squared Error): Regla que evalúa la función de pérdida a optimizar en un plano de coordenadas. Se obtiene a partir de la siguiente expresión:

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{j=1}^{N}(y_j - \hat{y})^2}$$

- **Mediana del Error Absoluto** (Median Absolute Error): Cuantifica el grado medio de la magnitud de errores, agnóstico a la dirección del error. Se obtiene a partir de la siguiente expresión:

$$\text{MAE} = \frac{1}{n}\sum_{j=1}^{N}|y_j - \hat{y}_j|$$

Tanto MAE como RMSE expresan el error promedio en términos del vector objetivo, con un rango del error

$$\varepsilon \rightarrow [0, \infty^{+})$$

Ambas métricas son agnósticas a la dirección de los errores:

- RMSE preserva la cardinalidad y evita la suma cero mediante la raíz cuadrada.
- MAE preserva cardinalidad mediante el valor absoluto.

Algunas reglas a considerar:

$$\text{MAE} > \text{(R)MSE}$$

Significa que el punto equidistante de los errores es superior a la métrica del error cuadrático promedio. En esta situación, esperamos tener observaciones atípicas con sobreestimaciones. Esta situación puede ocurrir cuando nuestro modelo falla en identificar valores atípicos. Esperamos tener una mayor concentración en valores bajos de la distribución de errores.

$$\text{MAE} < \text{(R)MSE}$$

Significa que el punto equidistante de los errores es inferior a la métrica del error cuadrático promedio. En esta situación, esperamos tener observaciones atípicas con subestimaciones. Esta situación puede ocurrir cuando nuestro modelo falla en identificar valores atípicos. Esperamos tener una mayor concentración en valores altos de la distribución del error.