**ElasticNet: Una técnica intermedia:**

Hasta el momento hemos podido ver dos técnicas de regularización en las cuales añadimos un componente de penalización en el proceso donde encontramos los valores de los parámetros 𝛽 minimizando la función de error.

Por ejemplo, si usamos el método de Mínimos Cuadrados Ordinarios, tenemos por definición nuestra función definida como:

Text, letter

Description automatically generated

Ahora bien. Si aplicamos la regularización L1 también conocida como Lasso (Least Absolute Shrinkage and Selection Operator), tenemos una ecuación de la forma:

Text, letter

Description automatically generated

donde tenemos un parámetro de ajuste llamado ƛ que si tiene valores altos para el problema mandará el valor de 𝛽j a 0.

Por otro lado. Si aplicamos la regularización L2 también conocida como Ridge, tendremos la siguiente ecuación:



Tendremos una penalización también pero que no tiene la posibilidad de llevar los valores de los coeficientes a cero. Sin embargo esto nos permitirá realizar el intercambio de +sesgo por -varianza.

Recordando que :

1. Ninguna de las dos es mejor que la otra para todos los casos.
2. Lasso envía algunos coeficientes a cero permitiendo así seleccionar variables significativas para el modelo.
3. Lasso funciona mejor si tenemos pocos predictores que influyen sobre el modelo.
4. Ridge funciona mejor si es el caso contrario y tenemos una gran cantidad.

Para aplicarlos y decidir cuál es el mejor en la práctica, podemos probar usando alguna técnica como cross-validation iterativamente. o bien, podemos combinarlos…

Regularización ElasticNet

Es común encontrarnos en la literatura con un camino intermedio llamado ElasticNet. Esta técnica consiste en combinar las dos penalizaciones anteriores en una sola función. Así, nuestra ecuación de optimización quedará:

Diagram

Description automatically generated with medium confidence

Donde tenemos ahora un parámetro adicional 𝛂 que tiene un rango de valores entre 0 y 1. Si 𝛂 = 0 , ElasticNet se comportará como Ridge, y si 𝛂 = 1 , se comportará como Lasso. Por lo tanto, nos brinda todo el espectro lineal de posibles combinaciones entre estos dos extremos.

1. Tenemos una forma de probar ambas L1 y L2 al tiempo sin perder información.
2. Supera las limitaciones individuales de ellas.
3. Si hace falta experiencia, o el conocimiento matemático de fondo, puede ser la opción preferente para probar la regularización.

ElasticNet con Scikit-learn

Para implementar esta técnica añadimos primero el algoritmo ubicado en el módulo linear\_model.

from sklearn.linear\_model import ElasticNet

Y luego simplemente lo inicializamos con el constructor ElasticNet() y entrenamos con la función fit().

regr = ElasticNet(random\_state=0)

regr.fit(X, y)

Al comparar con distintos valores de alpha, al parecer lo que nos indica es que es mejor idea usar Ridge que lasso.

# elasticnet model

**def** **modelElastic**(alpha=1):

modelElastic= ElasticNet(random\_state=0, alpha=alpha)

modelElastic.fit(X\_train, y\_train)

y\_predic\_elastic=modelElastic.predict(X\_test)

# loss function

elastic\_loss = mean\_squared\_error(y\_test, y\_predic\_elastic)

**return** elastic\_loss

alphas = np.arange(0,1,0.01)

loss\_total = []

**for** i **in** alphas:

res = modelElastic(i)

loss\_total.append(res)

loss\_total = np.array(loss\_total)

plt.plot(alphas, loss\_total)

plt.xlabel('alphas')

plt.ylabel('Loss Elastic')

plt.text(0.02, 0.8, 'loss min:{}'.format(np.min(loss\_total)), fontsize=7)

plt.show()

Chart, line chart

Description automatically generated

El modelo lineal te da incluso mucho mejor, aquí te dejo la comparación:

Error del modelo lineal: 7.141203442215841e-08

Error del modelo lasso: 0.0318013739963466

Error del modelo ridge: 0.0036293863527191326

El modelo lineal se disparo, es normal eso??

Al agregar este algoritmo de regularización al ejercicio anterior, vemos que tiene da por resultados que los otros dos y que lo que hizo fue lleva todos los coeficientes a cero.

Linear Loss: 9.11362706964361e-08

Lasso Loss: 0.04613142141272012

Ridge Loss: 0.005158990858509752

ElasticNet Loss: 1.1991244060565747

================================

Coef Linear: [[1.00012822 0.99987164 0.9999141 1.00013558 0.9996458 1.00007078

0.99987976]]

Coef Lasso: [1.35269253 0.86910384 0.37857477 0.83535984 0. 0.22914234

0.90887934]

Coef Ridge: [[1.08843372 0.95300018 0.84338277 0.90422291 0.64360573 0.76090281

0.96456701]]

Coef Elastic: [0. 0. 0. 0. 0. 0. 0.]

A continuación les dejo el código:

**import** pandas **as** pd

**import** sklearn

**from** sklearn.linear\_model **import** LinearRegression

**from** sklearn.linear\_model **import** Lasso

**from** sklearn.linear\_model **import** Ridge

**from** sklearn.linear\_model **import** ElasticNet

**from** sklearn. model\_selection **import** train\_test\_split

**from** sklearn.metrics **import** mean\_squared\_error

**if** \_\_name\_\_ == "\_\_main\_\_":

dataset = pd.read\_csv('./dataset/felicidad.csv')

# print(dataset.describe())

X = dataset[['gdp', 'family', 'lifexp', 'freedom',

'corruption', 'generosity', 'dystopia']]

y = dataset[['score']]

# print(X.shape)

# print(y.shape)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.25, random\_state=0)

modelLinear = LinearRegression().fit(X\_train, y\_train)

y\_pred\_linear = modelLinear.predict(X\_test)

modelLasso = Lasso(alpha=0.02).fit(X\_train, y\_train)

y\_pred\_lasso = modelLasso.predict(X\_test)

modelRidge = Ridge(alpha=1).fit(X\_train, y\_train)

y\_pred\_ridge = modelRidge.predict(X\_test)

modelElasticNet = ElasticNet(random\_state=0).fit(X\_train, y\_train)

y\_pred\_elastic = modelElasticNet.predict(X\_test)

linear\_loss = mean\_squared\_error(y\_test, y\_pred\_linear)

print('Linear Loss: ', linear\_loss)

lasso\_loss = mean\_squared\_error(y\_test, y\_pred\_lasso)

print('Lasso Loss: ', lasso\_loss)

ridge\_loss = mean\_squared\_error(y\_test, y\_pred\_ridge)

print('Ridge Loss: ', ridge\_loss)

elastic\_loss = mean\_squared\_error(y\_test, y\_pred\_elastic)

print('ElasticNet Loss: ', elastic\_loss)

print('='\*32, '\n')

print("Coef Linear: ", modelLinear.coef\_)

print("Coef Lasso: ", modelLasso.coef\_)

print("Coef Ridge: ", modelRidge.coef\_)

print("Coef Elastic: ", modelElasticNet.coef\_)```

\*\*