Tarea5_regu_

December 14, 2017

1 Tarea 5. Regularización.

L.E. Rojón
138442

In [1]:
```
import pandas as pd
import numpy as np
import random as rd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, mean_squared_error, r2_score
from scipy.stats import norm
from sklearn import preprocessing, linear_model
from random import random
import requests
import csv
from sklearn.naive_bayes import MultinomialNB
```

In [16]:
```
df = pd.read_csv('https://raw.githubusercontent.com/ClaseML-2017/MaterialyTareas/master
df.describe()
```

Out[16]:

	X	X2	X3	X4	X5	y
count	1029.000000	1029.000000	1029.000000	1029.000000	1029.000000	1029.000000
mean	48.052380	3113.669342	1.542203	-0.011250	628.000000	3113.490528
std	28.379936	2891.956247	0.452374	0.717016	297.191016	2892.963581
min	0.006314	0.000040	-2.199666	-0.999993	114.000000	0.000040
25%	23.174764	537.069692	1.365015	-0.731530	371.000000	522.757565
50%	47.655827	2271.077831	1.678116	-0.046938	628.000000	
75%	71.372550	5094.040864	1.853531	0.732296	885.000000	
max	99.873062	9974.628611	1.999448	1.000000	1142.000000	

```
50%     2262.728789
75%     5093.260718
max    10028.067820
```

```
In [17]: indexa=np.array([1 if random() < 0.75 else 0 for i in range(len(df))])
```

Las variables se guardan en arreglos independientes. Para que el fit pueda leer los datos hay que usar un reshape sobre ellos

```
In [18]: X_train= np.array(df[df.columns[0:-1]])[indexa==1]
         X_train = X_train.astype(float)
         X_test=np.array(df[df.columns[0:-1]])[indexa==0]
         X_test = X_test.astype(float)
         Y_train=np.array(df[[df.columns[-1]]])[indexa==1]
         Y_train = Y_train.astype(float)
         Y_test= np.array(df[[df.columns[-1]]])[indexa==0]
         Y_test = Y_test.astype(float)
         Y_test2 = Y_test
         X_test2 = X_test
         WisTrain = np.array([1.0 for i in range(0,X_train.shape[1])])
```

```
In [19]: from sklearn import preprocessing
         scaleX = preprocessing.StandardScaler()
         scaleY = preprocessing.StandardScaler()
         scaleX.fit(X_train)
         X_train = scaleX.transform(X_train)
         scaleY.fit(Y_train)
         Y_train = scaleY.transform(Y_train)
```

A continuación, el algoritmo para ajustar coeficientes. Recordar que es importante estandarizar.

```
In [22]: def ajustarCoef(x, y, w, aprend, lambd):

             w0 = 1
             error = 0.0
             YTemp = 0.0
             ErrorList=list()
             w = np.array([1.0 for i in range(0,X_train.shape[1])])

             for i in range(0,x.shape[0]):
                 error = y[i]-np.dot(x[i][:],w)-w0
                 ErrorList.append(error)
                 gradiente = error*aprend
                 w0 = gradiente + w0
                 for k in range(0,x.shape[1]):
                     w[k] =(gradiente*x[i][k]) - (lambd*w[k]) + w[k]

             return w0, w, ErrorList
```

```
        def reRun(n,lambd):
            for i in range(0,n):
                W0 = ajustarCoef(X_train, Y_train, WisTrain, 0.05, lambd)[0]
                W1 = ajustarCoef(X_train, Y_train, WisTrain, 0.05, lambd)[1]
                ErrorList = ajustarCoef(X_train, Y_train, WisTrain, 0.05, lambd)[2]

            return W0, W1, ErrorList




        X_test2=scaleX.transform(X_test2)
        Y_test2=scaleY.transform(Y_test2)




        def Testing(lambd):
            Test = reRun(50,lambd)
            W0_1000 = Test[0]
            W1_1000 = Test[1]
            error_sum = 0

            for i in range(len(X_test2)):
                y = np.dot(X_test2[i],W1_1000) + W0_1000
                error = (y - Y_test2[i])**2
                error_sum=error_sum+error

            return error_sum/len(X_test2)
```

In [23]:
```
        lambdaArray = np.array([0.0,0.01,0.02,0.07,0.09,0.11,0.13,0.15,0.17,0.19,0.21,0.23,0.25
        results_Array = np.array([1.0 for i in range(0,lambdaArray.shape[0])])
        for i in range(len(lambdaArray)):
            results_Array[i]= Testing(lambdaArray[i])
        results_Array
```

Out[23]:
```
array([ 1.59858185,  0.01788192,  0.03923313,  0.14278797,  0.18272597,
        0.22052307,  0.25573265,  0.28819136,  0.31791234,  0.3450084 ,
        0.36964288,  0.39200002,  0.41226789,  0.43062909,  0.44725598,
        0.46230856,  0.47593372,  0.48826546,  0.49942536,  0.50952341,
        0.5186588 ,  0.59031636])
```

In [24]:
```
        plt.plot(lambdaArray,results_Array)
```

```
plt.title('lambda vs. errores')
plt.show()
```