# Tarea6_Model_Selection

December 14, 2017

## 1 Tarea 6. Selección de Modelos.

L.E. Rojón
138442

```
In [2]: import pandas as pd
        import numpy as np
        import random as rnd
        import matplotlib.pyplot as plt
        from sklearn.model_selection import train_test_split
        from sklearn import linear_model
        from sklearn import preprocessing
```

Leemos los datos y hacemos la estandarización con standard scaler

```
In [3]: df = pd.read_csv('https://raw.githubusercontent.com/ClaseML-2017/MaterialyTareas/master/

        X_train, X_test, Y_train, Y_test = train_test_split(df.drop('y', axis=1), df[['y']], tra
        X_train, X_test, Y_train, Y_test = np.array(X_train), np.array(X_test), np.array(Y_train
        scalerX = preprocessing.StandardScaler().fit(X_train)
        X_train = scalerX.transform(X_train)
        X_test = scalerX.transform(X_test)
        scalerY = preprocessing.StandardScaler().fit(Y_train)
        Y_train = scalerY.transform(Y_train)
        Y_test = scalerY.transform(Y_test)
```

Construimos la función con la que entrenaremos el modelo y la función del error cuadrático medio.

```
In [7]: def out(w,X):
            return X.dot(w[1:]) +w[0]

        def train(X, y, w, la, eta = 0.01):
            for i in range(len(X)):
                err = y[i] - out(w, X[i])
                w[0] = w[0] + eta * (err)
                w[1:] = w[1:] + eta * (err*X[i]) - la * w[1:]
            return w
```

```
        def ECM(X, Y, w):
            return np.mean((X.dot(w[1:])+ w[0] - Y)**2)
```

Ahora construimos la función para hacer el muestreo de los subconjuntos y la función que hace la validación cruzada.

```
In [5]: def sub(X, k):
            I = np.random.choice(len(X), len(X), False)
            return [I[i] % k for i in range(len(X))]



        def crossvalidate(X_train, Y_train, w, k, la):
            S = np.array(sub(X_train, k))
            err = []
            for j in range(k):
                X_tr, Y_tr, X_tst, Y_tst = X_train[S != j], Y_train[S != j], X_train[S == j], Y_
                w = train(X_tr, Y_tr, w, la)
                err = np.append(err, ECM(X_tst, Y_tst.flatten(), w))
            err_prom = np.mean(err)
            return w, err_prom
```

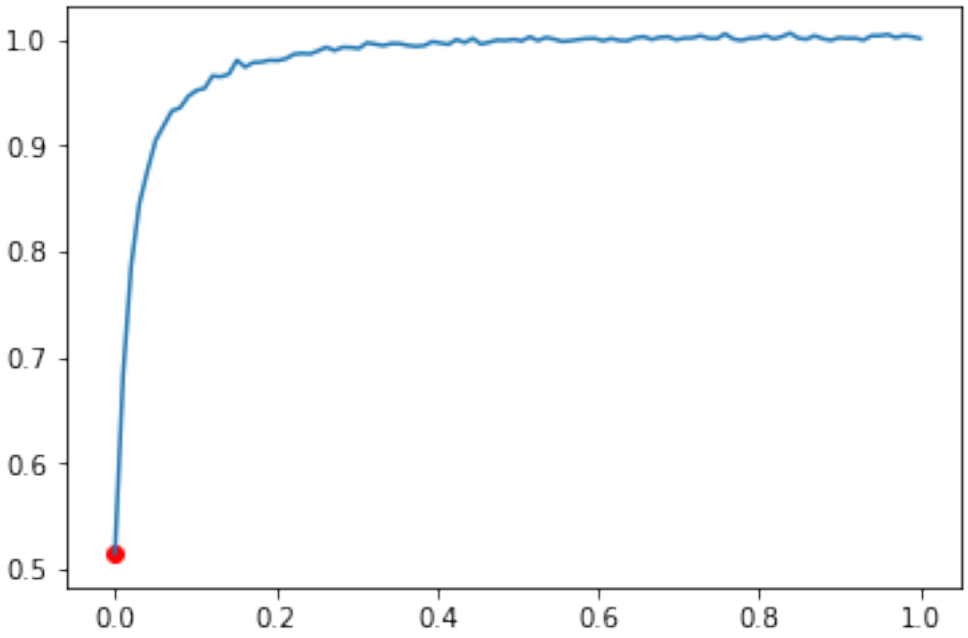
## 2 Prueba con datos

```
In [8]: la = np.linspace(0.0, 1.0, 100)
        err_prom = []
        for i in range(len(la)):
            w = np.asarray([rnd.random() for j in range(len(X_train[0]) + 1)])
            w, err = crossvalidate(X_train, Y_train, w, 10, la[i])
            err_prom = np.append(err_prom, err)

In [9]: err_prom

Out[9]: array([ 0.51544247,  0.6835088 ,  0.78699155,  0.84535766,  0.87655537,
                0.90522484,  0.91910282,  0.9329201 ,  0.93584781,  0.9465485 ,
                0.95222907,  0.95412151,  0.96599039,  0.96515043,  0.9676579 ,
                0.98074497,  0.97432926,  0.97861488,  0.97899864,  0.98079774,
                0.9804709 ,  0.98221287,  0.98642931,  0.98720752,  0.9867235 ,
                0.98977433,  0.99309218,  0.98990051,  0.99306025,  0.99285953,
                0.99187401,  0.99729764,  0.99594421,  0.99456905,  0.9966065 ,
                0.99644091,  0.99470976,  0.99389176,  0.99452226,  0.9983315 ,
                0.9970458 ,  0.99576582,  1.00025278,  0.99739904,  1.00141165,
                0.99606242,  0.99771934,  0.99979744,  0.99937476,  1.00047222,
                0.99902586,  1.00284923,  0.9993131 ,  1.00229248,  1.00064349,
                0.99847503,  0.9991581 ,  1.0003226 ,  1.00122306,  1.0013961 ,
                0.9991479 ,  1.00128606,  0.99945525,  0.9991601 ,  1.00198514,
```

```
                1.00305632,  1.00034444,  1.00240877,  1.00289659,  1.00013941,
                1.0016419 ,  1.00180863,  1.00399205,  1.00146672,  1.00117857,
                1.00572048,  1.00088709,  0.99952586,  1.00144651,  1.00189274,
                1.00399448,  1.0009347 ,  1.00308298,  1.00642831,  1.00164757,
                1.00057074,  1.00394029,  1.0011888 ,  0.99955472,  1.00211808,
                1.00143414,  1.0016385 ,  0.99976139,  1.00390056,  1.00400646,
                1.00505001,  1.00217308,  1.00409282,  1.00302001,  1.00125895])
```

```
In [10]: err_min = np.amin(err_prom)
         la_min = np.argmin(err_prom)
         print la[la_min], err_min

         plt.plot(la, err_prom)
         plt.scatter(la[la_min], err_min, c='r')
         plt.show()
```

```
0.0 0.515442472484
```




Finalmente, usamos la lambda óptima para entrenar.

```
In [11]: w = np.asarray([rnd.random() for i in range(len(X_train[0]) + 1)])
         for i in range(len(X_train)):
             w = train(X_train, Y_train, w, la[la_min])


         print ECM(X_train, Y_train.flatten(), w)
         print ECM(X_test, Y_test.flatten(), w)
```

```
0.474319349123
0.556560673382
```