TALLER DE REGRESIÓN USANDO SVM (KERNEL RBF Y POLY) CON SCIKIT-LEARN

El algoritmo de Máquinas de soporte vectorial para la clasificación contempla del uso de kernels para el uso con datos que no son linealmente separables. El siguiente script presenta el uso de 2 kernel, RBF y Polinomial, implementado con la librería scikit-learn de Python haciendo validación cruzada con K-pliegues.

Kernel RBF.

# Se importan las librerias:

from sklearn.model\_selection import cross\_val\_score

from sklearn.svm import SVC

import sklearn.metrics as mtr

import numpy as np

import the\_data as dt

# Se hace la division

X\_train, X\_test, y\_train, y\_test = dt.return\_data()

# se eligen varios valores de gamma: gamma es el parametron que determina el comportamiento de la regresion usando el kernel RBF

gamma = [10\*\*i for i in range(-3, 3)]

cs = [10\*\*i for i in range(-5,5)]

pos\_max\_score = np.array(0)

max\_scores = np.array(0)

# se prueban varios valores de C (parametron de regularización) y valores de gamma.

for gm in gamma:

cls\_list=[]

for c in cs:

cls = SVC(kernel='rbf', random\_state=0, gamma=gm, C=c).fit(X\_train, y\_train)

cls\_list.append(cls)

score = [cross\_val\_score(st, X\_train, y\_train, cv=30, n\_jobs=-1).mean() for st in cls\_list]

# pos of the best C's

pos\_max\_score = np.append(pos\_max\_score, np.argmax(score))

max\_scores = np.append(max\_scores, score[np.argmax(score)])

pos\_max\_score = np.delete(pos\_max\_score, 0)

max\_scores = np.delete(max\_scores, 0)

pos\_best\_gamma = np.argmax(max\_scores)

best\_gamma = gamma[pos\_best\_gamma]

best\_c = cs[pos\_max\_score[pos\_best\_gamma]]

svm\_valid = SVC(kernel='rbf', random\_state=0, gamma=best\_gamma, C=best\_c).fit(X\_train, y\_train)

model = svm\_valid.predict(X\_test)

f1 = mtr.f1\_score(y\_test,model)

recall = mtr.recall\_score(y\_test,model)

accuracy = mtr.accuracy\_score(y\_test,model)

precision = mtr.precision\_score(y\_test,model)

tn, fp, fn, tp = mtr.confusion\_matrix(y\_test, model).ravel()

specificity = tn/float(tn+fp)

Luego, se imprimen las métricas del modelo hallado.

Kernel polinomial (poly)

El script es similar al anterior, solo que los parámetros cambian. Con el kernel ‘poly’, se elige el grado de transformación que usará el algoritmo:

dg = [i for i in range(2,11)]

Y la linea para hallar los modelos, variando el grado y el parametro C es:

cls = SVC(kernel='poly', random\_state=0, degree=val, C=c).fit(X\_train, y\_train)

Las pruebas no fueron posibles debido a que el algoritmo tarda demasiado tiempo en ejecutarse.