Hw3 coding part

October 28, 2019

1 HW : 3 - Coding Part

Juan Vila

1.1 Part 3

[6]:
```
import numpy as np

def normalize(x):
    rv =  x/np.sqrt(np.dot(x.T,x))
    return rv

def find_zero(x):
    idx = np.argwhere(np.all(x[..., :] == 0, axis=0))
    rv = np.delete(x, idx, axis=1)
    return rv


def projection(U,X):
    p1 = np.dot(U.T,X)
    rv = np.dot(U,p1)
    return rv

def gs_algorithm(A):
    a_nonzero = find_zero(A)
    U = normalize(a_nonzero[:,0])
    A_j=a_nonzero[:,1:]
    n = np.shape(A_j)[1]
    for i in range(n):
        x_j= A_j[:,i]
        x_j_prime = x_j - projection(U,x_j)
        if x_j_prime.sum() == 0:
            continue
        U = np.c_[ U, normalize(x_j_prime)]

    return U
```

```
#checking with Part 2
x_2=np.array([[3,1],[0,3],[0,4]])
gs_algorithm(x_2)
```

[6]:
```
array([[1. , 0. ],
       [0. , 0.6],
       [0. , 0.8]])
```

1.2 Part 7

[12]:
```
import random
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt

iris=load_iris()
X = iris['data']
y = iris['target']


def beta_est(Y,x,cons=True):
    if cons:
        X = np.column_stack((x,np.ones([len(x),1])))
    else:
        X=x.copy()
    return np.dot(np.linalg.inv(np.dot(X.T,X)),np.dot(X.T,Y))

def proyection_2(Y,x, cons=True):
    if cons:
        X = np.column_stack((x,np.ones([len(x),1])))
    else:
        X=x.copy()
    y_hat = np.dot(X,beta_est(Y,x,cons))
    return y_hat

def normalization(X):
    rv = X.copy()
    for i in range(np.shape(X)[1]):
        mean = X[:,i].mean()
```

```
        sd = X[:,i].std()
        rv[:,i] = (X[:,i] - mean)/sd
    return rv


def cross_val(y,X,n_train, n_test, n):
    '''
    n_train: Size train set
    n_test: Size test set
    n: number of repetitions
    '''

    data = np.column_stack((y,X))

    #data_cv = np.split(data,n)
    rv =[]

    for i in range(n):
        np.random.shuffle(data)
        test_set  = np.array(random.sample(data.tolist(),n_test))
        train_set = np.array(random.sample(data.tolist(),n_train))
        y_test_orig, x_test_orig =  test_set[:,[0]], test_set[:,1:]
        y_train_orig, x_train_orig = train_set[:,[0]], train_set[:,1:]
        unique_train, counts_train = np.unique(y_test_orig, return_counts=True)
        unique_test, counts_test = np.unique(y_train_orig, return_counts=True)

        if len(unique_train) != 3 or len(unique_test) != 3:
            while len(unique_train) != 3 or len(unique_test) != 3:
                np.random.shuffle(data)
                test_set  = np.array(random.sample(data.tolist(),n_test))
                train_set = np.array(random.sample(data.tolist(),n_train))
                y_test_orig, x_test_orig =  test_set[:,[0]], test_set[:,1:]
                y_train_orig, x_train_orig = train_set[:,[0]], train_set[:,1:]
                unique_train, counts_train = np.unique(y_test_orig,␣
↪return_counts=True)
                unique_test, counts_test = np.unique(y_train_orig,␣
↪return_counts=True)

        count = 0
        for i in range(3):
            y_train = np.where(y_train_orig == i, 1, 0)
            y_test = np.where(y_test_orig == i, 1, 0)
            w = beta_est(y_train, x_train_orig)
            x_test = np.column_stack((x_test_orig,np.
↪ones([len(x_test_orig),1])))
            y_hat = np.dot(x_test,w)
            y_label_assig = np.where(y_hat>.5,1,0)
```

```
            for i,j in enumerate(y_test):
                if y_label_assig[i] != y_test[i]:
                    count+=1

        rv.append(count/(3 * n_test))
    rv=np.array(rv)

    return rv.mean()
```

1.2.1 7b

[13]:
```
random.seed(1234)
x_norm = normalization(X)
cross_val(y,x_norm,40,10,100)
```

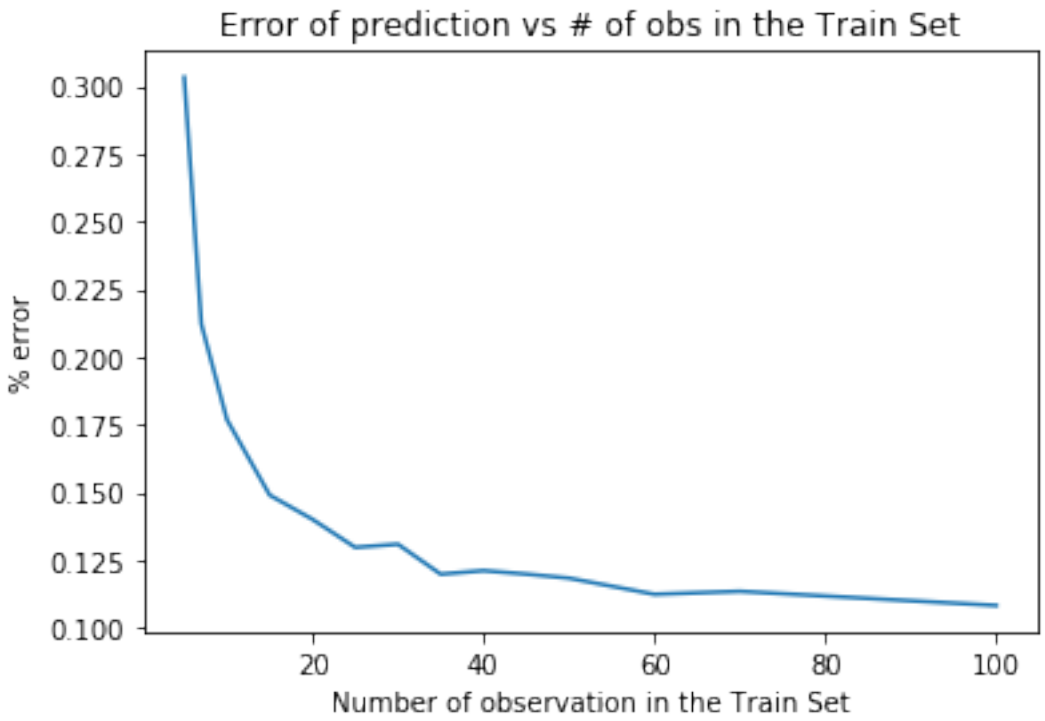
[13]:
```
0.13333333333333333
```

Comment: We can see that error is 13.33% on average of a trainning set of 40 obs and a test set of 10 obs, with 100 repetitions.

1.2.2 7c

[14]:
```
random.seed(1234)
train_nobs_set = [5,7,10,15,20,25,30,35,40,45,50,60,70,100]
test_result = []
for i in train_nobs_set:
    test_result.append(cross_val(y,x_norm,i,10,1000))
train_nobs_set=np.array(train_nobs_set)
test_result=np.array(test_result)
plt.plot(train_nobs_set,test_result)
plt.ylabel('% error')
plt.xlabel('Number of observation in the Train Set')
plt.title('Error of prediction vs # of obs in the Train Set')
```

[14]:
```
Text(0.5, 1.0, 'Error of prediction vs # of obs in the Train Set')
```



Comment: we can see as the size of the train set increase the error diminish.

1.2.3 7d

[15]:
```
x_d = x_norm[:,[0,1,2]]
random.seed(1234)
cross_val(y,x_d,40,10,100)
```

[15]:
```
0.13899999999999998
```

Comment: We see that if we run the model only with the three first variables the error increase.