Parcial

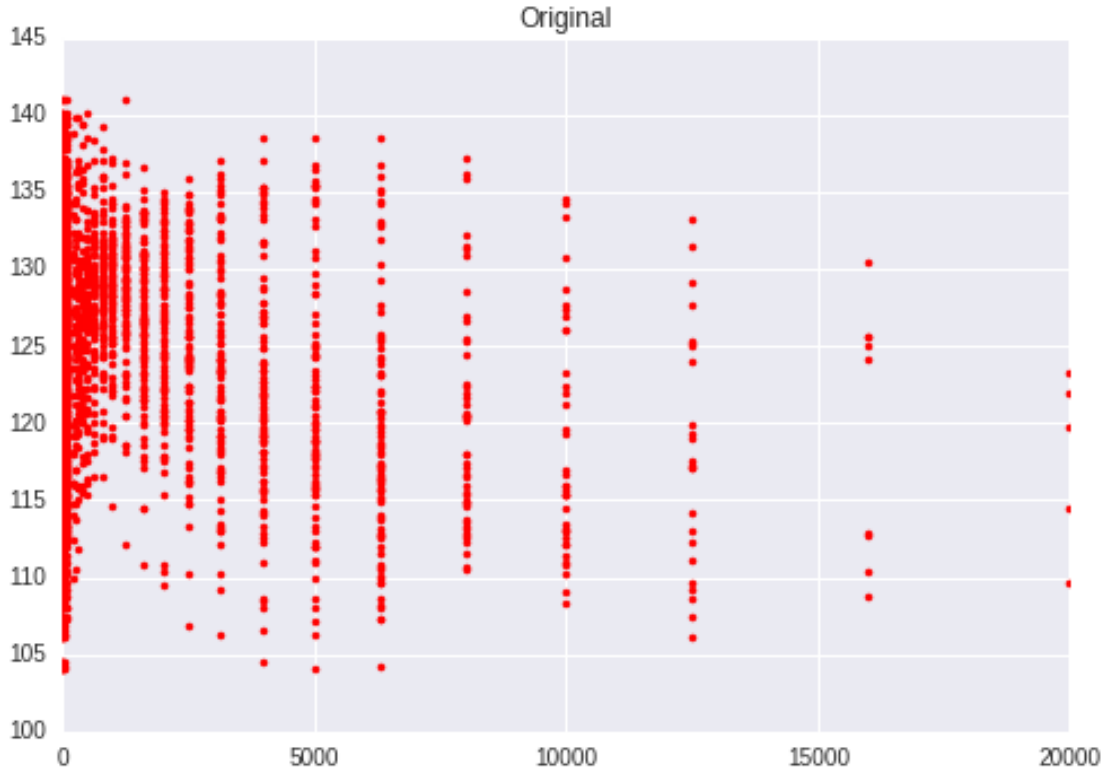
September 27, 2016

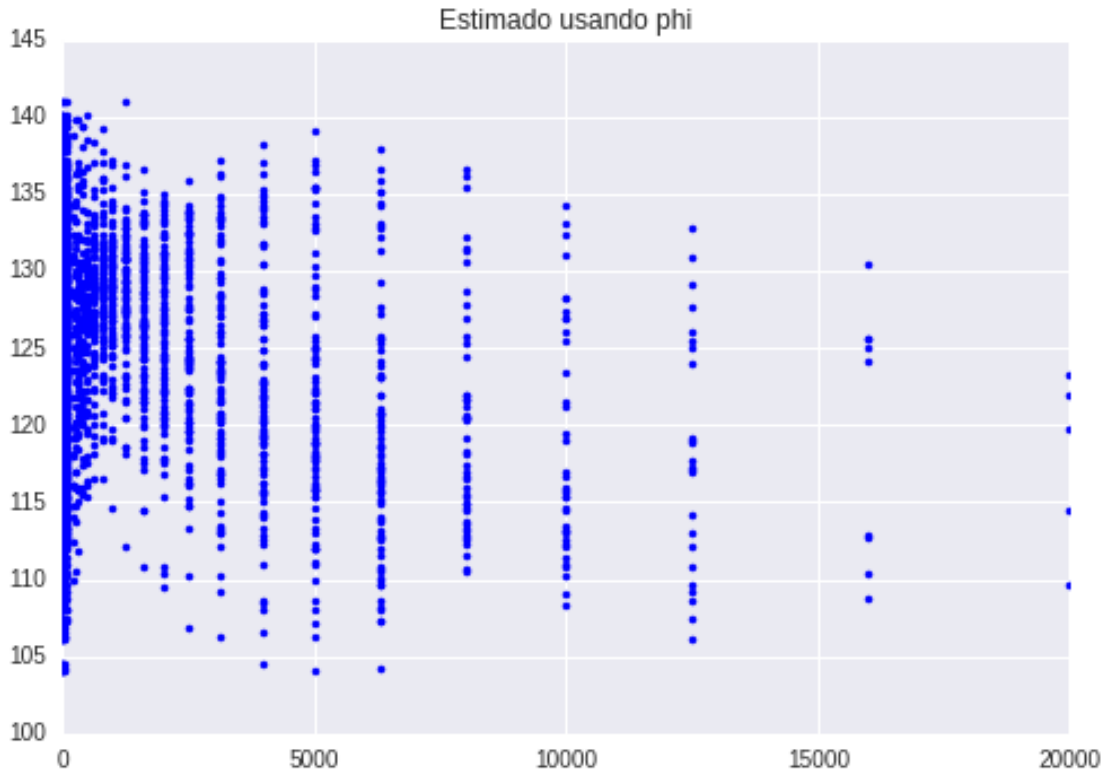
In [1]:
```
%matplotlib inline
import numpy as np
import scipy as sp
import matplotlib as mpl
from matplotlib import pyplot as plt
from numpy.random import multivariate_normal as mnormal_rd
from scipy.optimize import lsq_linear
import pandas as pd
import seaborn as sns
from scipy import io
```

In [2]:
```
data = io.loadmat('./parcial/P1/datosPrueba1.mat')

ytrain = data['ytrain']
xtrain = data['Xtrain']
xtest = data['Xtest']

def lineal_least_squares(X, y, l = 0.0):
    C = X.transpose().dot(X)
    s = X.shape[1]
    T = np.linalg.pinv(C + l * np.eye(s, s))
    w =  T.dot(X.transpose()).dot(y)
    ye = X.dot(w)
    return [ye, w, C]

def comp_phi(X1, X2, sig=0.5):
    n = X1.shape[0]
    m = X2.shape[0]
    phi = np.zeros((n, m))
    for i in range(n):
        for j in range(m):
            d = X1[i] - X2[j]
            phi[i][j] = np.exp(-(d.T.dot(d)) / (2 * sig))
    return phi

phi = comp_phi(xtrain, xtrain)
print (phi)
```

```
        ye, w, C = lineal_least_squares(phi, ytrain)

        plt.plot(xtrain, ytrain, 'r.')
        plt.title('Original')
        plt.show()
        plt.plot(xtrain, ye, 'b.')
        plt.title('Estimado usando phi')
        plt.show()
```

```
[[ 1.  0.  0. ...,  0.  0.  0.]
 [ 0.  1.  0. ...,  0.  0.  0.]
 [ 0.  0.  1. ...,  0.  0.  0.]
 ...,
 [ 0.  0.  0. ...,  1.  0.  0.]
 [ 0.  0.  0. ...,  0.  1.  0.]
 [ 0.  0.  0. ...,  0.  0.  1.]]
```

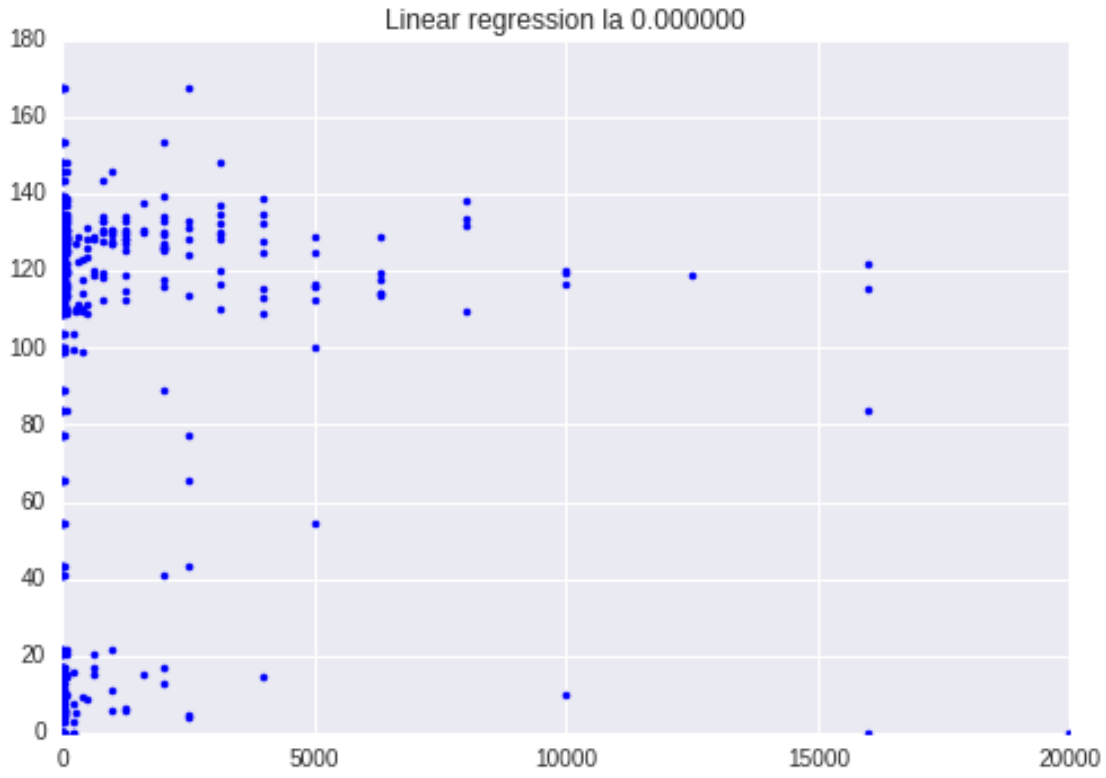


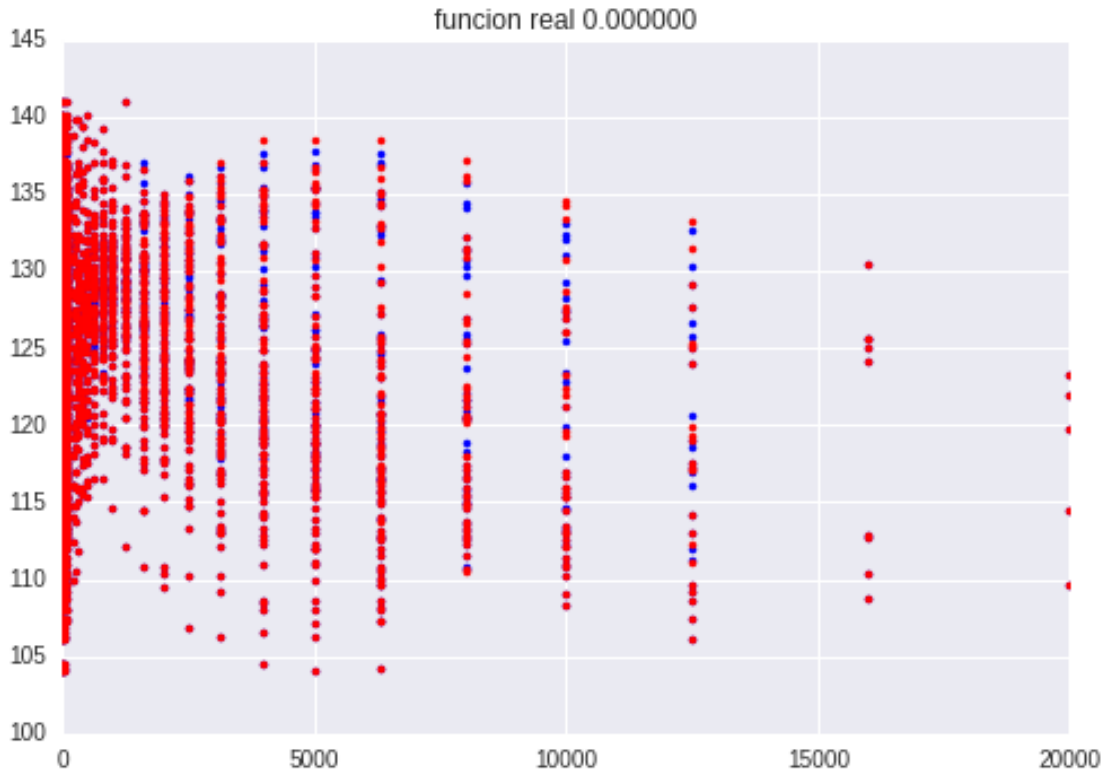


In [3]:
```
data = []

def simlsq(la, sig=0.5, save=False):
    phi = comp_phi(xtrain, xtrain, sig)
    ye, w, C = lineal_least_squares(phi, ytrain, la)

    error = np.sum((ye - ytrain) ** 2)
    print ('Error:', error, ' la ', la, ' sig ', sig)
    nX = comp_phi(xtest, xtrain, sig)
    print (nX.shape)

    plt.title('Linear regression la %2f' % (la))
    plt.plot(xtest, nX.dot(w), 'b.')
    np.savetxt('./punto1.txt', nX.dot(w))
    data.append(nX.dot(w))
    plt.show()
    plt.title('funcion real %2f' % (la))
    plt.plot(xtrain, phi.dot(w), 'b.')
    plt.plot(xtrain, ytrain, 'r.')
    plt.show()

simlsq(0, True)
```

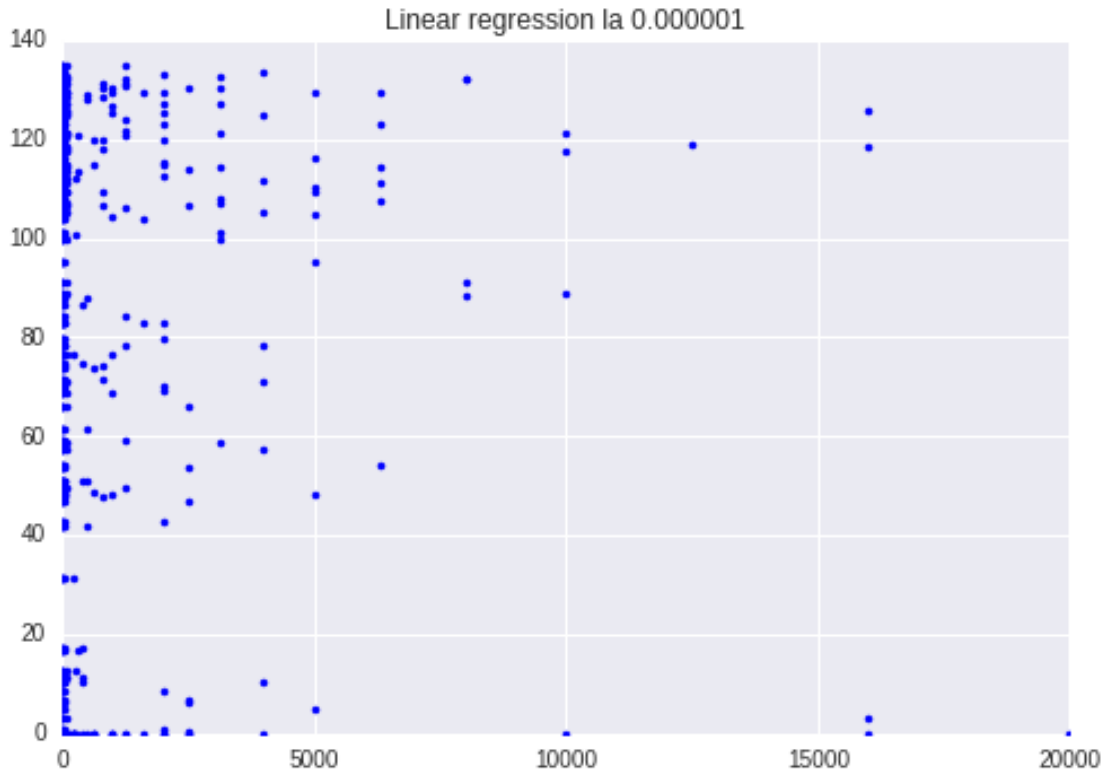
```
simlsq(0.000001, 0.2)
```

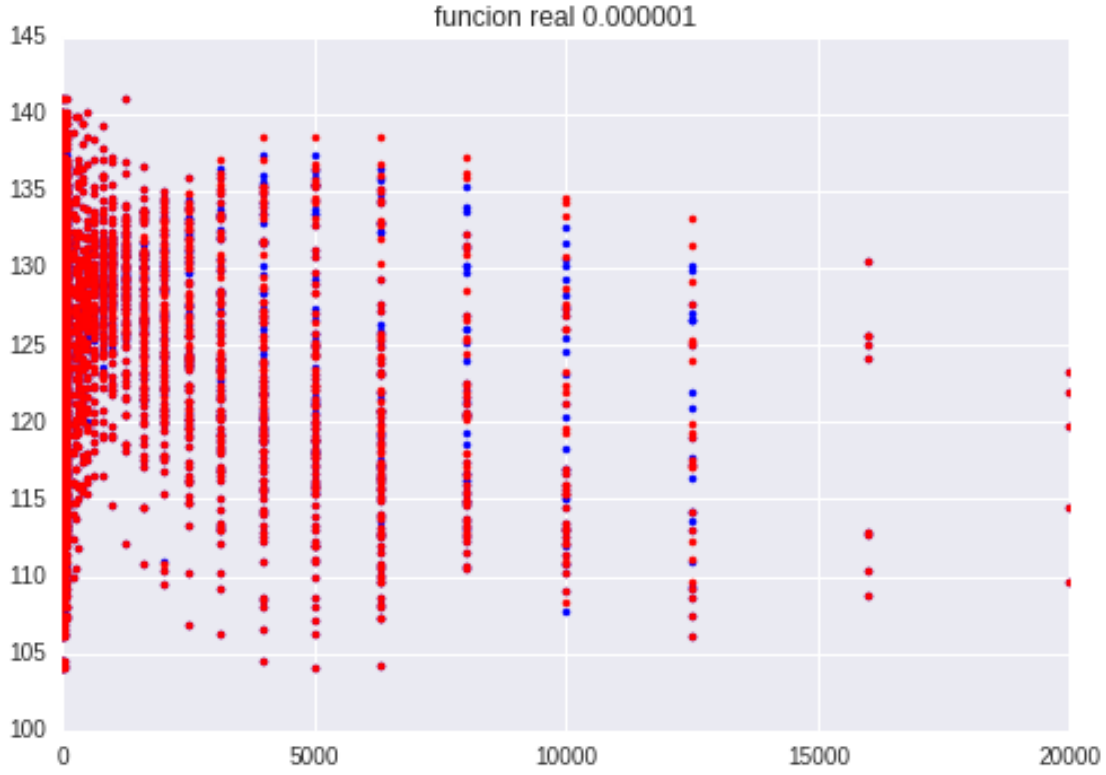
```
Error: 495.595855281  la  0  sig  True
(150, 1353)
```



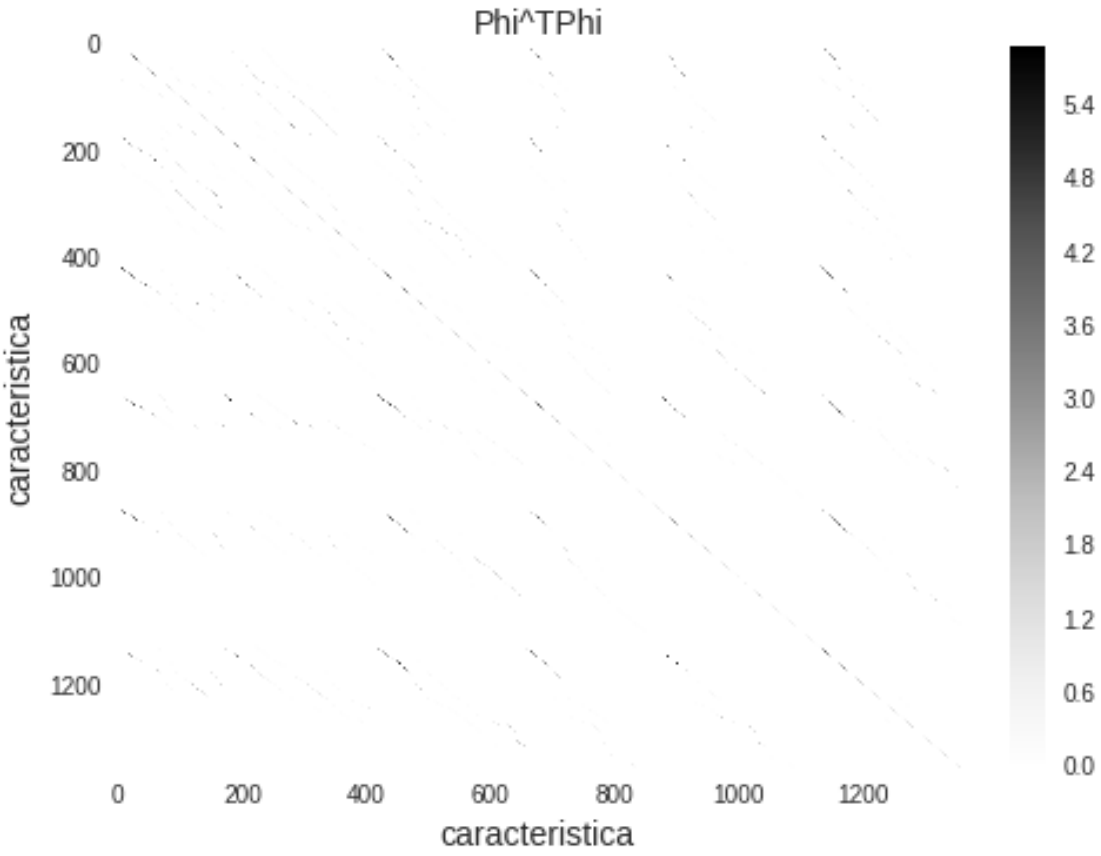



```
Error: 381.688923387  la  1e-06  sig  0.2
(150, 1353)
```


Linear regression la 0.000001
140
120
100
80
60
40
20
0
0
5000
10000
15000
20000


funcion real 0.000001
145
140
135
130
125
120
115
110
105
100
0
5000
10000
15000
20000

In [4]:
```
C = phi.T.dot(phi)
plt.imshow(C, aspect='auto')
plt.colorbar()
plt.xlabel('caracteristica', fontsize = 14)
plt.ylabel('caracteristica', fontsize = 14)
plt.title('Phi^TPhi', fontsize = 14)
plt.show()
```



In [5]:
```
data = io.loadmat('./parcial/P2/datosPrueba2.mat')

ltrain = data['ltrain']
xtrain = data['Xtrain']
xtest = data['Xtest']

print (xtrain.shape)
```

```
def perceptron(X, Y, w_init, rate=0.1, fun=None, iters=100, stop=None):
    w = w_init
    w_ant = None
    for it in range(0, iters):
        w_ant = w
        for i in range(0, len(X)):
            if (np.sign(w.dot(X[i])) != Y[i]):
                w = w + rate * X[i] * Y[i]
        if (stop):
            if (stop(w, w_ant)):
                break

    if (fun):
        fun(X, Y, w)
    return w

def print_err(X, Y, w):
    ytag = np.array([w.dot(X[i]) for i in range(len(X))])
    ytag = np.sign(ytag)
    print('Misclassified %.2f' % (100 * np.sum(ytag != Y) / X.shape[0]))


def sim(w, eta, save=False):
    wtag = perceptron(xtrain, ltrain[:,0], w, rate=eta, iters=1000, fun=pri
    ytag = np.array([wtag.dot(xtrain[i]) for i in range(len(xtrain))])

    ysol = np.array([wtag.dot(xtest[i]) for i in range(len(xtest))])
    ysol = np.sign(ysol)
    print ('Xtest', xtest.shape)
    print (ysol)
    if (save):
        np.savetxt('./punto2.txt', ysol)


# Start at zero
w = np.zeros(xtrain.shape[1])
eta = 0.01

sim(w, 0.01, True)
sim(w, 1)
sim(w, 0.001)

# Start random

w = np.random.rand(xtrain.shape[1])
eta = 0.01

sim(w, 0.01)
```

```
        sim(w, 1)
        sim(w, 0.001)
```

```
(584, 784)
Misclassified 0.00
Xtest (64, 784)
[ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
Misclassified 0.00
Xtest (64, 784)
[ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
Misclassified 0.00
Xtest (64, 784)
[ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
Misclassified 0.00
Xtest (64, 784)
[ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
Misclassified 0.00
Xtest (64, 784)
[ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
Misclassified 0.00
Xtest (64, 784)
[ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
```

```
In [6]: def sim_phi(eta, save=False):
            x = comp_phi(xtrain, xtrain)
            w = np.zeros(x.shape[0])
            wtag = perceptron(x, ltrain[:,0], w, rate=eta, iters=1000, fun=print_er
            ytag = np.array([wtag.dot(x) for i in range(len(x))])
```

```
            phi_test = comp_phi(xtest, xtrain)
            ysol = np.array([wtag.dot(phi_test[i]) for i in range(len(phi_test))])
            ysol = np.sign(ysol)
            print ('X', x.shape)
            print (ysol)
            if (save):
                np.savetxt('./punto2_phi.txt', ysol)

        sim_phi(2, 0.01)
```

```
Misclassified 0.00
X (584, 584)
[ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
```